May 2018

GEORAC: an RNA-seq Atlas Constructor for the Gene Expression Omnibus

Aurash A. Mohaimani
University of Wisconsin-Milwaukee

GEORAC: AN RNA-SEQ ATLAS CONSTRUCTOR

FOR THE GENE EXPRESSION OMNIBUS

by

Aurash Mohaimani

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Biomedical and Health Informatics

at

The University of Wisconsin-Milwaukee

May 2018

ABSTRACT

GEORAC: AN RNA-SEQ ATLAS CONSTRUCTOR
FOR THE GENE EXPRESSION OMNIBUS

by

Aurash Mohaimani

The University of Wisconsin-Milwaukee, 2018
Under the Supervision of Professors Timothy B. Patrick and Rebecca D. Klaper

The meteoric rise of next-generation sequencing technologies over the past 15 years has resulted in a voluminous amount of data generated by modern biological and clinical studies. RNA sequencing, colloquially referred to as RNA-Seq, is a next-generation approach capable of surveying and quantifying whole organism transcriptomes. RNA-Seq methods are valued over microarray assays for their ability to avoid cross-hybridization signal noise, to quantify gene or transcript expression without assay-specific upper limits, to natively provide single-nucleotide genomic resolution, and to allow for de novo transcriptome assemblies. Many thousands of RNA-Seq studies have been published over the past seven years, and a significant area of bioinformatics research has focused on the creation of atlases that aggregate RNA-Seq results. These atlases are crucially useful for surveying trends in gene expression across published studies, for inspecting potentially contentious claims made by novel or prior work, and for synthesizing future research directions. The Expression Atlas currently serves as the canonical example for an RNA-Seq atlas and presents results from over 3,000 studies across numerous model research organisms.

An issue with the Expression Atlas is that it forcibly applies a uniform secondary re-analysis pipeline to each RNA-Seq study incorporated within its database; this approach presents a conceptual challenge to studies whose results have been generated and published using established, well-tested workflows. Thus, there exists a critical need to provide for construction of RNA-Seq atlases that precisely reflect original results presented within the literature, and the primary objective of this dissertation is to provide a workflow that allows for transparent, reproducible construction of RNA-Seq atlases from study meta- and expression data housed within the National Center for Biomedical Information's Gene Expression Omnibus (GEO). The challenge of this goal is exacerbated by the highly flexible design of GEO, which allows researchers to define novel metadata attributes and values at will and to submit expression results in virtually any format.

Following an introductory background into modern genomics and RNA-Seq, the second chapter of this work presents GEOMP, a metadata parser and relational database constructor for the Gene Expression Omnibus. The subsequent third chapter describes GEOMP2, an in-place augmentation of GEOMP that provides further atomization and loading of sample-specific characteristics tags; this chapter significantly presents results from a pilot study surveying bioinformatics methods reproducibility across the zebrafish, mouse, and human research communities using metadata parsed and output by GEOMP2. Chapter four details GEORGET, a pipeline designed to rehabilitate, translate, and load expression data pulled from GEO into the relational database store constructed by GEOMP2. Chapter five concludes with discussion of future directions needed to expand and improve upon the current GEORAC workflow and the associated methods reproducibility study.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This work would not have been possible without assistance and guidance from many individuals. First, special thanks are due to the co-advisors of this dissertation, Dr. Tim Patrick and Dr. Rebecca Klaper; the former of these two critically taught me that informaticians are meta-scientists [rather than being fundamental scientists] foremost, while the latter was the first professor to critically recognize and acknowledge my potential for growth as a doctoral candidate. Significant additional thanks are owed to the members of this dissertation's committee: Dr. Michael Carvan, who provided perspective and motivation for the underlying spirit of this work; Dr. Susan McRoy, who provided helpful advice and comments for shaping future manuscript submissions resulting from this work; and Dr. Elizabeth Worthey, who continually reminded me that bioinformatics applications ultimately aim to empower biological researchers and clinicians alike.

My time as a candidate has been divided across three sets of work groups, with special thanks owed to members from each group for imparting their advice and guidance over the years. From the Laboratory for Public Health Informatics and Genomics within the Zilber School of Public Health: Dr. Chun-Yuan Huang, the research scientist who first introduced me to applied bioinformatics; Dr. Kourosh Ravvaz, my longtime friend and mentor in science; and Charles Murphy, who advised that one consider "reading everything before doing anything". From the Rat Genome Database group within the Medical College of Wisconsin: Alexander Stoddard, for sharing invaluable bioinformatics expertise and domain knowledge; Marek Tutaj, who provided mentorship on the art of effective programming; Jeff De Pons, for enlightening me on the tree-like nature of Web document design; and Jennifer Smith, for her faithfully exhaustive quality control and assurance of my previous work. Additional thanks are extended to members of the RGD curation team, many of whom encouraged my progress from afar.

Most recently, I have served as the sole bioinformatician of the Great Lakes Genomics Center within the School of Freshwater Sciences, and tremendous thanks are owed to the individuals that have facilitated my successful time here: Dr. Raymond Hovey, for his prior mentorship and discussions regarding the nature of bioinformatics work, and Angie Schmoldt, for her perpetually sunny disposition and persistent optimism. Great thanks are also owed to the University's High Performance Computing service team: Jason Bacon, who has taught and mentored me in the art of Unix computing, effective parallel programming, and life; Daniel Siercks, for his continual empathy regarding the challenges of this experience; and David Crass, for imparting upon me the inclination to appreciate unsavory experiences. Auxiliary thanks are needed for the human resources personnel who have invaluably assisted me along this journey: Tanika Reesnes, Debra Abanathy, Mallory Kaul, Nicole Oake, and Nina Ottman.

This dissertation is dedicated to my loving family: my mother, Bea; my father, Mohaiman; my brother, Arya; and my beloved, Amanda. Darling, you and B.B. are my guiding stars!

Chapter 1: Introduction

Next-Generation Sequencing

The 1977 publication and rapid subsequent adoption of the Sanger method for DNA sequencing [12], itself being an evolution of the preceding "plus-and-minus" method [13], heralded the beginning of the genomics era, catapulting early bioinformatics past protein sequencing and early structural analysis into the processing of whole organism genomes [14]. The Sanger method employs a catalyzed enzyme reaction, in which DNA polymerase elongates a strand complementary to the DNA strand of primary sequencing interest—the template strand. Mechanistically, the template strand often possesses a DNA marker region whose nucleotide sequence is known, and a primer is attached to the complementary strand that hybridizes to the template's marker; polymerase will then elongate the complementary strand, beginning from the end of the primer. Throughout elongation, polymerase employs an ambient pool of fluorescently-labeled dideoxynucleotides, each lacking a hydroxyl group that would otherwise enable continuous multi-nucleotide extension of the complementary strand. In this way, variable-length single nucleotide elongation of the complementary strand becomes sequentially observable in the Sanger method. [14]

A variant of the Sanger method, colloquially known as "shotgun sequencing", emerged in the early 1980s through the work of S. Anderson [15]. In this method, genomic DNA is stochastically fragmented into many smaller pieces—typically on the order of several kilobases in size—before being inserted into a bacterial vector and undergoing replication *in vitro*. The DNA is then amplified over several rounds before being exhaustively sequenced. Due to the random nature of sequence generation in this method, significant overlap will exist between sequenced fragments, and the challenge of matching overlapping fragments to create more comprehensive sequences birthed the bioinformatics

specialty of sequence assembly. The shotgun method typically sequences the same nucleotide up to [or well in excess of] ten times in a single experiment, and the degree of sequence overlap factors into the cost of this method. [14] Augmented variations of shotgun sequencing allowed Venter *et al* to sequence whole organism genomes, for which their draft of the human genome was published in 2001 [16].

Large-scale automation of the Sanger sequencing method allowed for viable conception and completion of the HGP well-ahead of the project's estimated timeline; the public consortium behind this gargantuan effort announced completion of their draft sequence in 2000 and published associated findings of their results in 2001 [17]. Although the HGP came in under-budget by its completion, the project was allocated funds on the order of several billion US dollars—such a sum could not possibly constitute a working basis for standardization and delivery of personalized genomic medicine well into the 21st century. Herein lies the critical weakness of the Sanger method, which has since been referred to as a "first-generation" sequencing technique—the Sanger approach requires vectorization and subsequent amplification of targeted DNA, making it both time-consuming and prohibitively costly [18, 19].

Introduction of second-generation sequencing platforms in the mid-2000s (Costa *et al* assert that massively parallelized sequencing platforms were first introduced in 2004), also known as "next-generation" sequencers, revolutionized the direction of genomics following completion of the HGP [19]. M. L. Metzker likened the introduction and potential application of next-generation techniques to the beginning years following initial publication of the polymerase chain reaction (PCR); use of next-generation sequencing platforms would be primarily limited by the imagination and vision of its users [18]. The arguably greatest advantage of next-generation sequencing platforms concerns their ability to generate billions of sequencing reads at low cost and relatively high speed—while the Sanger method was ultimately bottlenecked by its need to employ bacterial cloning, next-generation methods avoid this

costly and time-consuming pitfall altogether by using various forms of template, primer, or polymerase immobilization that enable the simultaneous occurrence of billions of sequencing reactions [18, 19].

The explosive growth of genomics through use of next-generation sequencing platforms over the past decade has resulted in a myriad of bioinformatics specialties: through DNA-seq, it has become possible to re-sequence targeted portions of select genomes, allowing for deeper genomic understanding of species of interest; DNA-seq has also permitted the advancement of comparative phylogenomics on a previously unachieved scale. With RNA-seq methods, new analysis pipelines permitting expression profiling across experimental samples have been developed, allowing for greater sensitivity to differential gene expression and noise reduction than ever previously possible with microarray assays [19-22]. Both DNA- and RNA-seq additionally lend well to *de novo* assembly of whole organismal genomes or transcriptomes. Finally, bioinformatics techniques developed within the last half-decade have facilitated the analysis of whole organism epigenomes, particularly with respect to DNA methylation, allowing for the tracking of DNA methylation changes across multiple generations of a target species, such as zebrafish. [23]

Next-Generation Sequencing: Platform Mechanics

Before delving further into the theoretical underpinnings of RNA-seq analysis pipelines and their associated output(s), it is helpful to briefly survey the mechanics for which many modern next-generation sequencing (NGS) platforms rely upon. Metzker details that, despite various platform mechanical differences regarding their actual operation (such as comparison of Roche's 454 line of pyrosequencers versus those of Illumina's sequence-by-synthesis, which typically employ solid-phase amplification), several key steps remain identical between the platform archetypes. Notwithstanding the final step of sequence data analysis—a topic to be revisited—the first two major steps entail template preparation and sequencing / imaging. [18]

It is paramount that template preparation involves non-biased fragmentation of nucleic acid, so as to ensure that a representative sample of genomic data is sequenced; a common theme amongst NGS platforms involves the immobilization of the template, such that billions of sequencing reactions may proceed in parallel. To achieve the number of template molecules needed for these billions of reactions, one of two prototypical template formation and immobilization strategies may be employed: clonal amplification of the template or single molecule use of the template. From the former, two specific methods tend to be used in order to amplify the template—emulsion PCR (emPCR) and solid-phase amplification. [18]

Emulsion PCR is designed to occur in a cell-free environment and avoids the arbitrary loss of genomic data, thusly providing a significant advantage over previous bacterial cloning-dependent methods. In emPCR, an oil-aqueous solution houses bead-DNA complexes, each of which possesses many universal primer attachment sites. Template molecules bind to the primers, becoming part of the bead complex, and elongation via ambient polymerase and dNTPs proceeds. Following subsequent template dissociation, beads will possess thousands of copies of the template sequence; the beads can then be chemically cross-linked to a glass slide, from which further NGS sequencing steps may be performed. [18]

Solid-phase amplification represents the second method within the clonal amplification strategy of template immobilization, in which randomly distributed forward- and reverse-oriented primers are covalently bound to a glass slide; surface density of the subsequently amplified clusters is largely determined by the ratio of primers to template strands. Two essential steps occur in solid-phase amplification: initial priming (e.g. immobilization) and extension of the single-stranded template molecule, followed by subsequent bridge amplification of the template to adjacent primers, forming clusters. The solid-phase amplification method can produce spatially separated template clusters that

number in the hundreds of millions, allowing for free template ends to be bound by a universal sequencing primer that initiates an NGS reaction. [18]

The other template immobilization strategy is composed of a number of methods related to single-molecule use of the template and is typically better suited for handling very small amounts of starting genomic material (at or less than a microgram of mass). Because single-molecule methods do not rely on PCR-based amplification, they tend to be less procedurally cumbersome and also benefit from the distinct advantage of avoiding spurious and incorrect DNA replication errors, including unintended amplification of AT- and GC-rich regions. Resultantly, quantitative NGS methods such as RNA-seq—which rely heavily on accurate, representative expression of genomic sequence—benefit particularly from single-molecule approaches. Three general methods are available within the single-molecule strategy, each involving the covalent binding (e.g. immobilization) of primers, template molecules, or polymerase, respectively, to the supporting material. [18]

Figure 1: Methods from Template Immobilization Strategies [18]

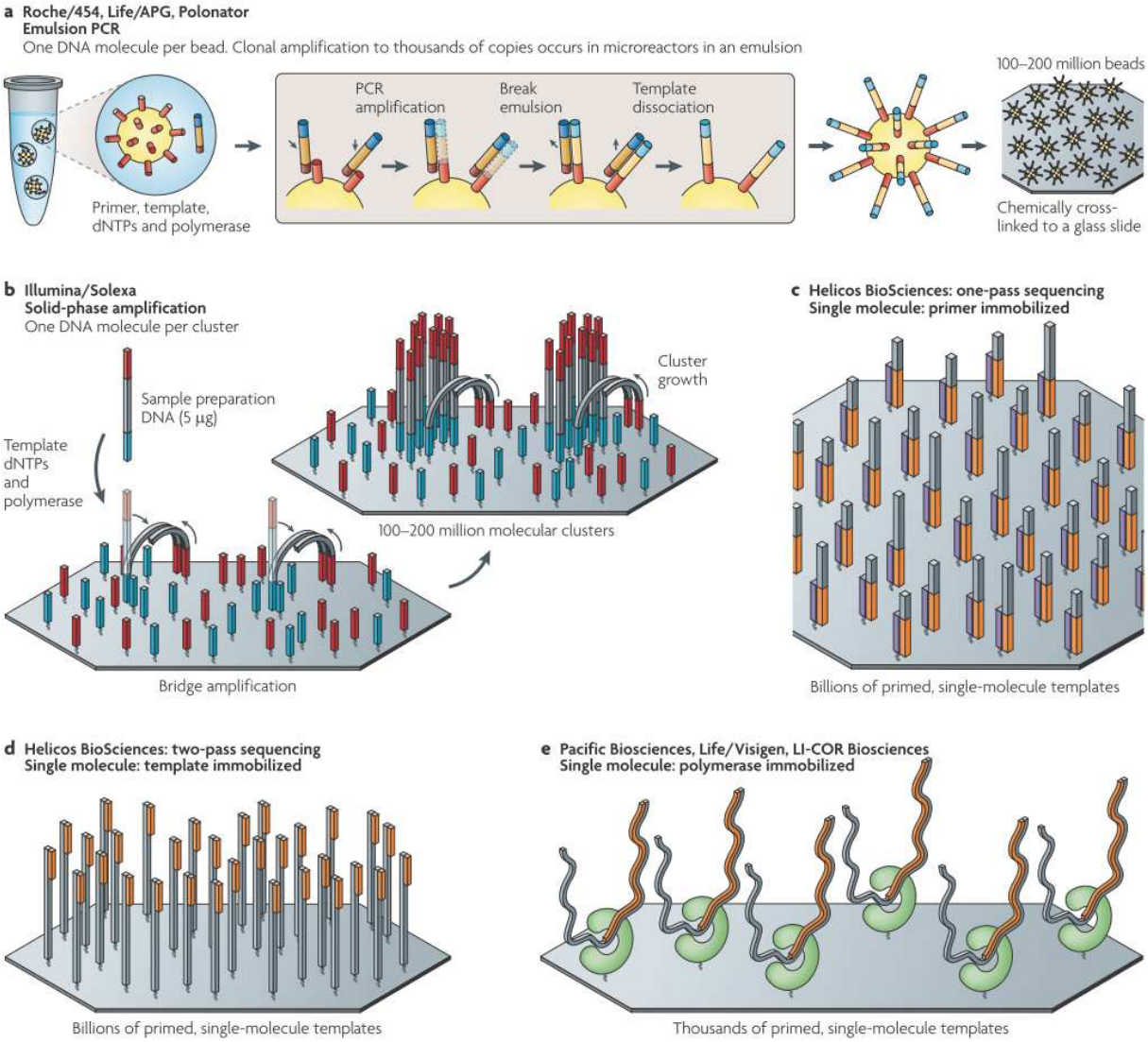


Figure 1: an illustration of each method within the two strategies of clonal amplification and single-molecule template use for immobilizing the template in NGS platforms. In **a**, emulsion PCR is demonstrated through use of template-bound primer bead complexes within an oil-aqueous solution; template dissociation allows for polymerase-facilitated creation of bead complexes possessing hundreds of millions of copies of template sequence for use in an NGS reaction. In **b**, solid-phase amplification is demonstrated through binding of template molecules to primers covalently bound to the support surface; bridge amplification across the surface enables the growth of clusters, which promotes further template synthesis for use in NGS reactions. Parts **c, d,** and **e** respectively demonstrate support-bound primers, template molecules, and polymerase for use in single-molecule reactions to generate immobilized template as part of the single-molecule strategy.

Following template immobilization, NGS platforms proceed with sequencing and imaging of genomic data—this area contains a number of diverse methods, each of which may participate in concert with one or both of the umbrella strategies designed for template immobilization. In cyclic reversible termination (CRT), a reversible terminator is bound to the primed template via polymerase, incorporating a single additional nucleotide to the sequence whilst halting additional nucleic acid replication. After washing off the ambient pool of unincorporated nucleotides, imaging techniques—which combine fluorescence and laser imaging—are used to read the incorporated nucleotide prior to cleavage of the inhibiting terminal group and final subsequent washing. This process is repeated cyclically for the length of the template, and immobilization of hundreds of millions of template molecules allows for the massive parallelization via CRT within NGS platforms. Specifically, two CRT specialty methods are employed by Illumina and Helicos BioSciences to address both template immobilization strategies: Illumina employs four-color CRT in conjunction with solid-phase amplified template molecules, a member of the clonal amplification strategy, while the HeliScope platform utilizes sequential single-color CRT in conjunction with the single-molecule template strategy. [18]

Figure 2: Methods of Cyclic Reversible Termination [18]

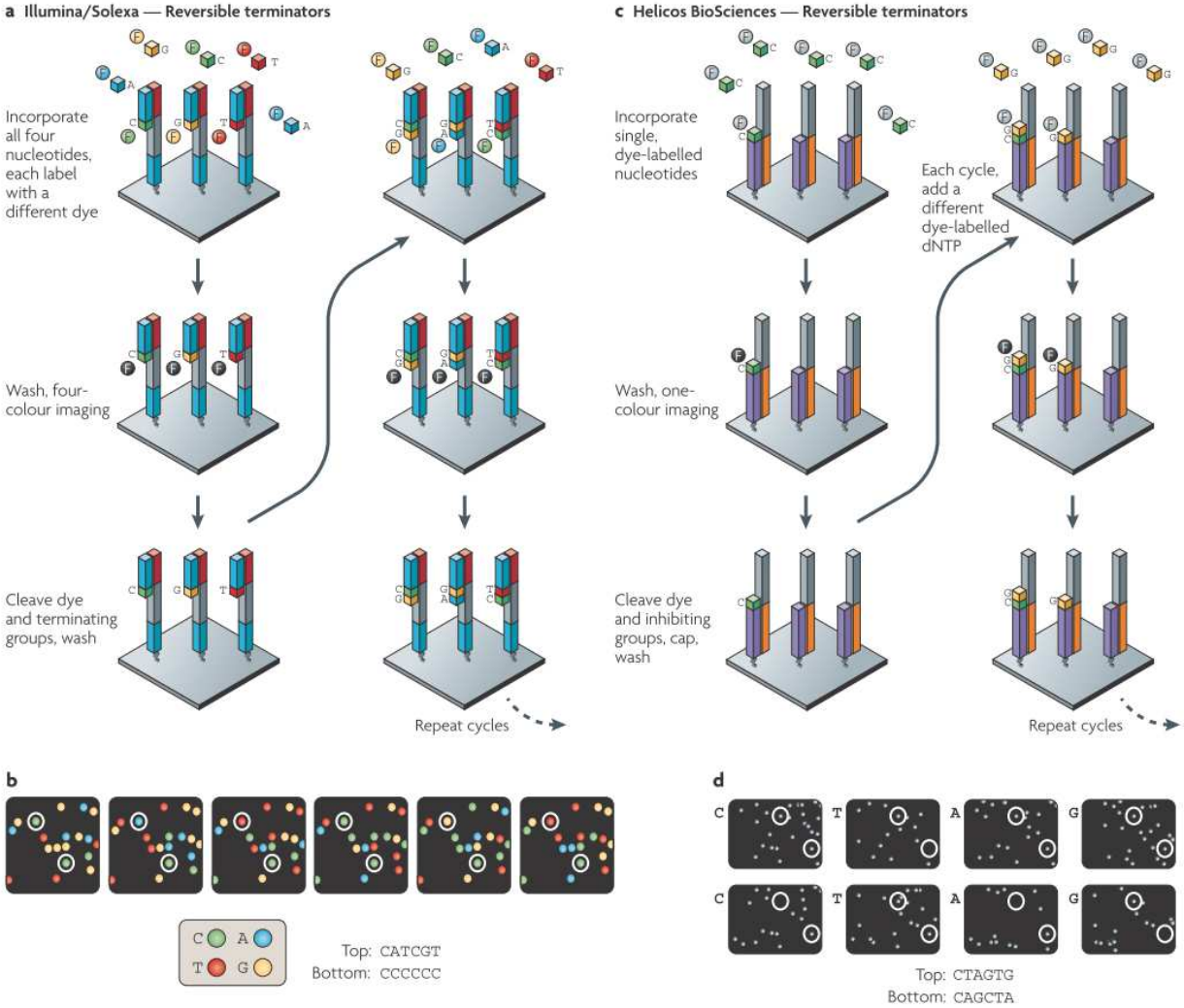


Figure 2: a depiction of the two cyclic reversible termination (CRT) methods found in modern NGS platforms marketed by Illumina and Helicos Biosciences. In **a** and **b**, the Illumina four-color CRT is illustrated, in which each cyclic round involves labeling via four-color dye solution that is subsequently washed away prior to imaging of the newly added nucleotide, cleavage of the terminal group, and additional washing. Each CRT cycle adds exactly one nucleotide to the complement of the template strand. In **c** and **d**, the HeliScope platform's single-color CRT is shown, where a single-color solution is presented to the immobilized template; the solution is washed away prior to nucleotide imaging, and subsequent cleavage of the dye and inhibiting terminal groups, followed by final washing, completes each cycle. Although each single-color CRT cycle adds at most one nucleotide, synthesized strands will often "dephase" with respect to each other over the course of multiple cycles.

An entirely separate method from CRT involves use of DNA ligase and is referred to as sequencing by ligation (SBL). Essentially, SBL employs a fluorescently labeled probe that hybridizes to the complementary strand adjacent to the primed, immobilized template. When DNA ligase is added, it joins the colored probe to the primer on the template strand, allowing for ambient unattached probes to be washed away prior to fluorescent imaging and sequencing. Cyclic rejuvenation in SBL can be achieved through use of cleavable probes to remove the fluorescent dye or through removal and subsequent hybridization of a new primer to the template strand with each cycle. Life/APG (now part of Applied Biosystems, a subsidiary of Thermo Fisher Scientific) has designed an entire NGS platform around the concept of SBL, known as support oligonucleotide ligation detection (SOLiD). The SOLiD platform functions through use of two-base encoded probes, offering improved accuracy in calls on colored nucleotides and single nucleotide variants (SNVs), the latter of which is made possible through inspection of valid adjacent color changes. Template immobilization for the SOLiD platform is achieved through emulsion PCR, a member of the clonal amplification strategy. [18]

Figure 3: Sequencing by Ligation in the SOLiD Platform [18]

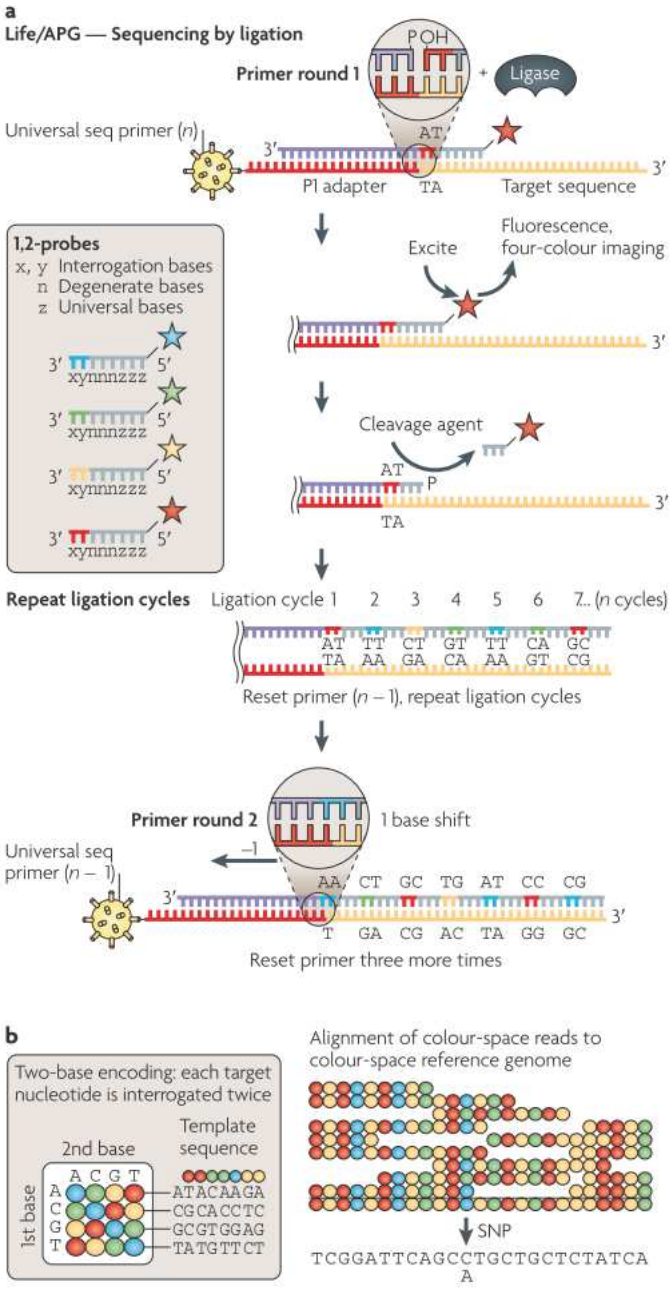


Figure 3: sequencing by ligation (SBL) in the SOLiD platform. In **a**, the four-color ligation method of the SOLiD platform is shown, with two-base probes being interrogated in-between three-base degenerate nucleotides. When the ligation cycle is shifted *(n – 1)*, another interrogation cycle is repeated—this continues for a total of ten times, though at minimum five *(n – 1)* shifts are necessary to reveal the target sequence. In **b**, the $4^2 = 16$ two-base color combinations are displayed, with homogenous two-base-pair windows always being colored in blue. Thus, if ligation involves prior knowledge regarding the first several nucleotides over

> multiple cycles, it quickly becomes feasible to determine subsequent nucleotide identity via two-base colored probes.

The final NGS platform methodology discussed at length by Metzker concerns pyrosequencing, a technique employed by Roche in their 454 platform, which was arguably the first commercially offered production-grade NGS platform released in 2004. Pyrosequencing is a non-electrophoretic, biochemically facilitated method that uses enzymatic reactions to release inorganic pyrophosphate as light. Where CRT and SBL employ modified nucleotides to halt nucleic acid synthesis, pyrosequencing makes use of limiting additions of dNTP to pause and resume the action of polymerase. Throughout controlled synthesis of the template strand's complement, light measured from the release of inorganic pyrophosphate is recorded in the form of flowgrams, which are used to determine the identity of newly added nucleotides. Template immobilization is prepared for the 454 pyrosequencing platform via emulsion PCR. [18]

Figure 4: Pyrosequencing with the Roche 454 Platform [18]

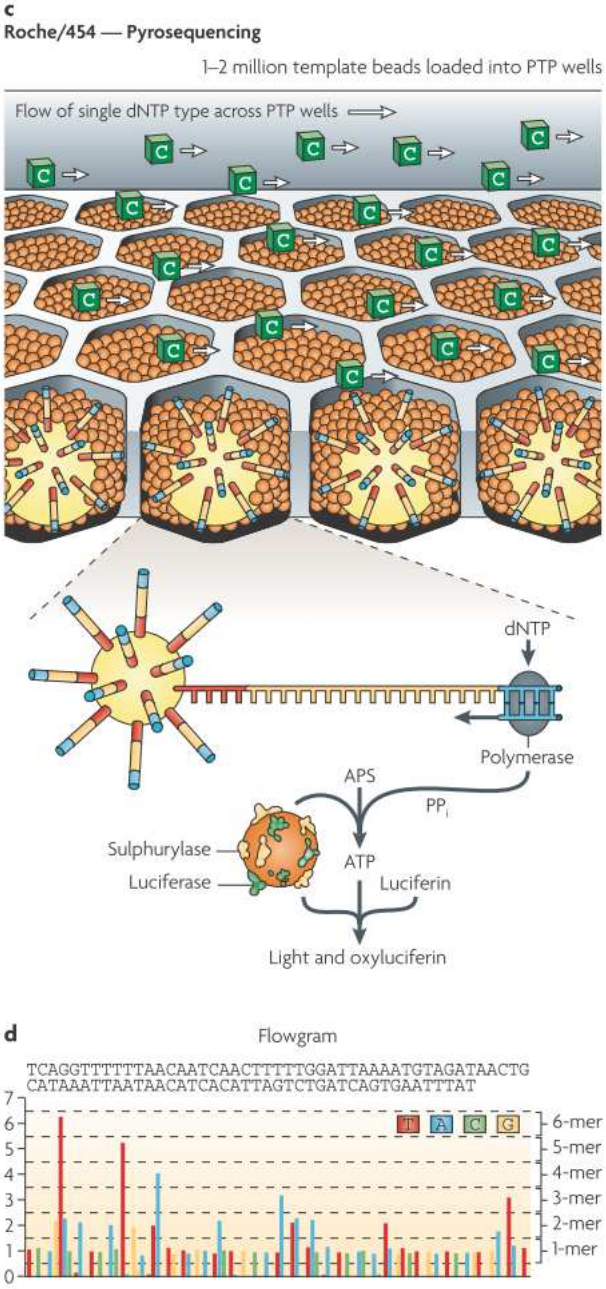


Figure 4: pyrosequencing with the Roche 454 platform. In **c**, an example cycle is shown: emPCR amplified DNA beads are added into wells, along with smaller beads containing sulphurylase and luciferase. As a single dNTP solution is run across the wells—cytosine in this example—the dNTP functions in conjunction with polymerase to catalyze a series of enzymatic reactions that convert inorganic pyrophosphate into signals of light. In **d**, the recorded light signals have been converted into a flowgram, which is used to determine nucleic acid sequence identity.

The past half-decade has borne witness to the rise of what some now term as “third-generation” and “fourth-generation” sequencing platforms, given that the four previously discussed NGS platforms are now being considered as “second-generation” sequencers [18, 24]. Regarding third-generation sequencers, E. E. Schadt, S. Turner, and A. Kasarskis alternatively describe these platforms as “real-time sequencers” (RT-seq), in which single template molecules are *not* amplified prior to sequencing by polymerase elongation. As this method avoids PCR amplification entirely, real-time sequencers—of which Pacific Biosciences has since marketed commercially [18]—promise to avoid errors that accompany PCR, such as amplification bias and strand de-phasing. By avoiding the time constraints for amplification, RT-seq platforms also promise to drastically reduce sequencing time, on the order of days to hours or even minutes. Despite these promises, third-generation sequencers (TGS) have exhibited a tendency to fall short of the bioinformatics community’s expectations, as they suffer from sequence identity inaccuracy ultimately stemming from a lack of sufficient passes around their generated circular polymerase reads. [24]

Fourth-generation sequencers are a novel technology best publicized by Oxford Nanopore, whose sequencing platform aims to employ a proprietary nanopore voltage gating technology to sequence small amounts of template material through sub-microscopic changes in voltage potentials as the target sequence is read. Like PacBio, the nanopore platform has suffered from a number of key challenges, most prominently a critical concern regarding sequence identity inaccuracy rates and reproducibility of sequencing experiments between different platform operators. [25] Both PacBio and nanopore sequencing platform technologies represent an exciting foray into the future of next-generation sequencing, and their mechanics lie outside the scope of this work; further exposition on these topics is left to the reader.

Overview of RNA-Seq

Before proceeding into any lengthy exposition on modern RNA-Seq methods and their advantages over the microarray assays they have superseded, it is essential to consider the historical methods that long preceded these modern techniques; these approaches continue to hold relevance in current, limited-scale gene quantification studies. The Northern blot technique, originally designed in the late 1970s, is typically employed to survey expression of small interfering RNAs (siRNAs) or micro RNAs (miRNAs) and initially involves size-oriented separation through use of denaturing polyacrylamide gel electrophoresis (dPAGE). The RNA molecules are subsequently transferred to a membrane, one that is usually composed of nylon, and are bound to the membrane via cross-linking facilitated by ultraviolet (UV) radiation. Though use of UV radiation is rapid and inexpensive, it has been suggested that UV results in an overabundance of cross-linking, hindering downstream results; Pall and Hamilton have proposed a chemically-based alternative for performing the cross-linking step of Northern blotting, resulting in more than an order of magnitude improvement to the sensitivity of identifying and quantifying small RNA expression. [41]

The explosive proliferation of the polymerase chain reaction (PCR) technique throughout the early 1990s and 2000s has profoundly altered the landscape of molecular biology and remains a dominant technique to this day, seeing continued frequent use within countless next-generation sequencing library preparation protocols. PCR derivatives, such as quantitative PCR (qPCR) and reverse transcriptase PCR (rtPCR), were developed in the early 1990s to quantify gene expression from samples as small as a single cell. However, a well-noted shortcoming of initially designed rtPCR reactions was the need to simultaneously observe amplification of the samples up to a certain point, the plateauing point, within the log phase of the reactions; preparing samples in this way can be laborious in the extreme. Heid *et al* proposed real-time quantitative PCR (RT-qPCR) as an answer to this issue: through use of selected “housekeeping” genes, one may start concurrent RT-qPCR reactions for genes of interest

without concern over when certain samples reach their plateauing point. [42] Vandesompele *et al* later proposed a geometric averaging model for further standardizing internal controls of genes submitted within the real-time qPCR methodology [43]. While this method and Northern blotting continue to demonstrate effectiveness for small-scale studies, the succeeding passages will expound upon the advantages of using more exhaustive approaches, such as RNA-Seq.

As a key specialty within bioinformatics, the study and performance of RNA-seq primarily concerns the transcriptome of a given biological system or organism, which is described by Z. Wang, M. Gerstein, 2and M. Snyder to be the complete set and number of transcripts found within a cell, given a certain development stage and physiological condition. The purpose behind RNA-seq studies is to generally gain greater understanding of whole organismal transcriptomes, given a myriad of experimental conditions. Greater comprehension of the transcriptome allows for a better understanding of physiological development and the course of genetic disease. [20]

From Crick's Central Dogma of molecular biology, it is well-known that DNA is transcribed into RNA prior to being translated into protein macromolecules; during this process, the immediately transcribed DNA—as whole (e.g. total) RNA—is spliced into a form known as messenger RNA (mRNA) that is subsequently translated. Total RNA is generally composed of two kinds of nucleic acid junctions, introns and exons, where introns represent non-coding regions of genes, while exons represent coding DNA to be translated into protein. Thus, multiple exonic junctions are typically spliced together to form the mRNA coding sequence required for translation into strings of amino acids that are folded and post-translationally modified to form mature proteins. Regions of nucleic acid that span the distances between genes are often referred to as intergenic regions. [1, 3, 19]

Contextually, the working definition of a gene in bioinformatics unfortunately suffers from a certain degree of ambiguity in the literature: because genes are composed of exonic junctions that are spliced together to form variations of a given gene in response to a slew of environmental conditions, a

gene may be considered to behave somewhat like a mathematical set, possessing a finite number of members. Each member constitutes an isoform of the gene, much like isotopes of given elements from the Periodic Table. A subset of these isoforms, those of which occur most commonly for a given gene and under homeostatic conditions, may be considered to be the transcript for the select gene, while abnormally occurring or disease-related isoforms of the gene may be considered as alternative isoforms. [19, 20, 22]

Historically, gene expression studies have been executed through use of microarrays, a platform designed to allow for the simultaneous observation of many gene expression levels within an entire cell line at once [19-22]. While powerful, microarray methods have ultimately suffered from a number of key shortcomings, foremost of which concerns the presence of significant background noise in every experiment's analysis, originating from the occurrence of cross-hybridization between various well samples in the tiling array [19, 20]. Secondly, microarrays typically feature poor resolution at the single nucleotide level, often requiring specialty techniques to correct via usage of pre-existing knowledge regarding the given genomic region of interest [20]. Finally, microarrays experience difficulties with determining large ranges of dynamic expression for given genes between experimental samples—that is, an up-regulation of a given gene on the order of hundreds or thousands of times between experimental samples tends to be very difficult to accurately quantify by microarray techniques [19, 20, 22].

The introduction of next-generation sequencing platforms and techniques to gene expression studies, colloquially referred to as RNA-seq analysis, has quickly become one of the most advanced specialties within bioinformatics over the past decade [19]. RNA-seq circumnavigates or improves upon the critical shortcomings of microarray approaches: because there is no use of a tiling array in RNA-seq experiments, cross-hybridization cannot occur, effectively eliminating background noise [19, 20]. Additionally, RNA-seq allows for the interrogation of genes at single-base resolution, allowing for a

definitive understanding of individual base changes, such as single nucleotide polymorphisms (SNPs) [19-21]. Finally, RNA-seq is not limited in the determination of dynamic gene expression levels; once corrected and normalized via statistical methods, aligned reads from RNA-seq experiments can reliably and reproducibly detect changes in gene expression levels that may differ by orders of magnitude [19, 20, 22].

A particular newfound strength associated with RNA-seq studies, which was generally not possible to achieve in traditional microarray experiments, is the potential capability to detect novel, undiscovered isoforms of genes. When combined with the aforementioned capacity to interrogate exonic single base-pair identity, it becomes possible for comprehensive RNA-seq studies to determine alternative transcripts of known or newly proposed genes. [19, 20] RNA-seq data is also not merely limited to differential gene expression analysis across samples within an experiment; data from these studies can also be employed to create *de novo* transcriptome assemblies of species for which their reference sequence does not yet exist, or for given genomic regions of interest in which the existing reference is inaccurate or incomplete [19-22].

Despite its many strengths, RNA-seq grapples with a number of key computing challenges: outside the realm of template preparation and sequencing, data analysis in RNA-seq requires rigorously-tested computational methods to ensure normalization of stochastic read alignment and reproducibility of analyses produced from initial experiment results. Examples of these methods take the form of a variety of read alignment algorithms, in-sample normalization methods, and between-sample differential expression analysis packages. Canonical pipelines that interweave the input and output from various RNA-seq related tools—whose construction, maintenance, and modification are non-trivial—are detailed below. [20, 21]

RNA-Seq Analysis Pipelines

A. Oshlack, M.D. Robinson, and M. D. Young describe the prototypical pipeline for RNA-seq analysis in terms of key steps and methodological components: first, millions of [short] reads are gathered and aligned against a reference sequence, which can be a whole genome (often for model organisms) or a transcriptome, which itself can either be known or assembled *de novo*. Following alignment, counts of reads in given genomic regions of interest, such as exonic junctions, genes, or coding regions are tabulated and statistically normalized so as to remove biases present within the initial dataset that stem from the stochastic nature of read alignment in next-generation sequencing platforms. Then, tables of resulting data—which can exist in either statistically normalized in-sample form or in terms of raw read counts—are input into a differential expression (DE) analysis package. The type of input given to the DE tool depends on the underlying model employed to determine differential expression of genomic features. Once differential expression results are obtained, for which gene expression values can be safely compared between samples (e.g. between-sample normalization), the list of significantly differentially expressed genes can be piped through a gene annotation toolkit to generate biological insight regarding the metabolic, physiological, and pathogenic pathways particularly involved with the set of experimental samples. [26]

Figure 5: Overview of RNA-seq Analysis Pipelines [26]

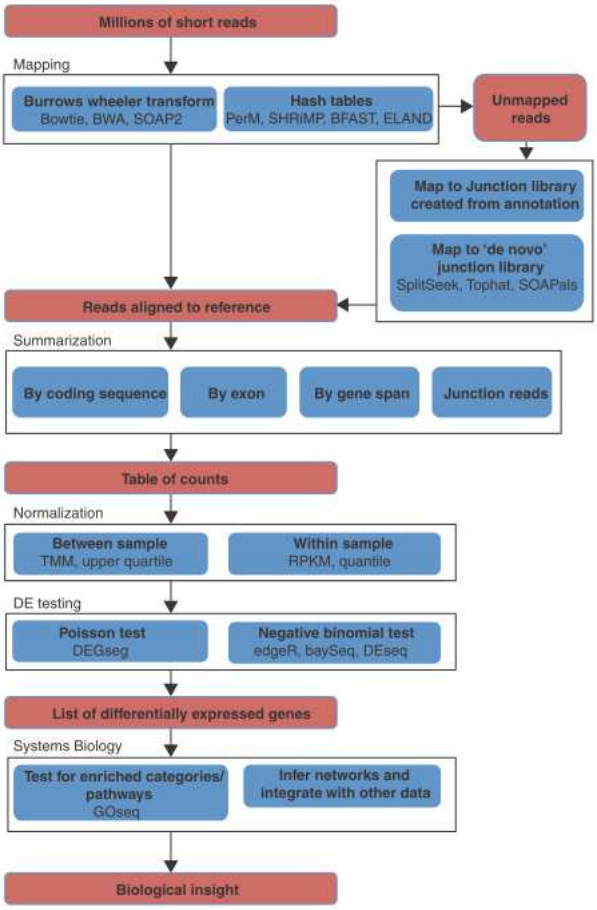


Figure 5: an overview of the general steps involved in performing analysis of RNA-seq data. Major pipeline steps are colored in red, while methodology components are shown in blue. Each methodology component is listed with example analysis packages or software used to achieve its respective aim; these lists are not intended to be comprehensive.

V. Costa *et al* provide an alternate, more generalized perspective on RNA-seq pipelines, with particular focus on the output of both short read alignment algorithms and read quantification methods. [19]

Figure 6: A Generalized RNA-seq Pipeline [19]

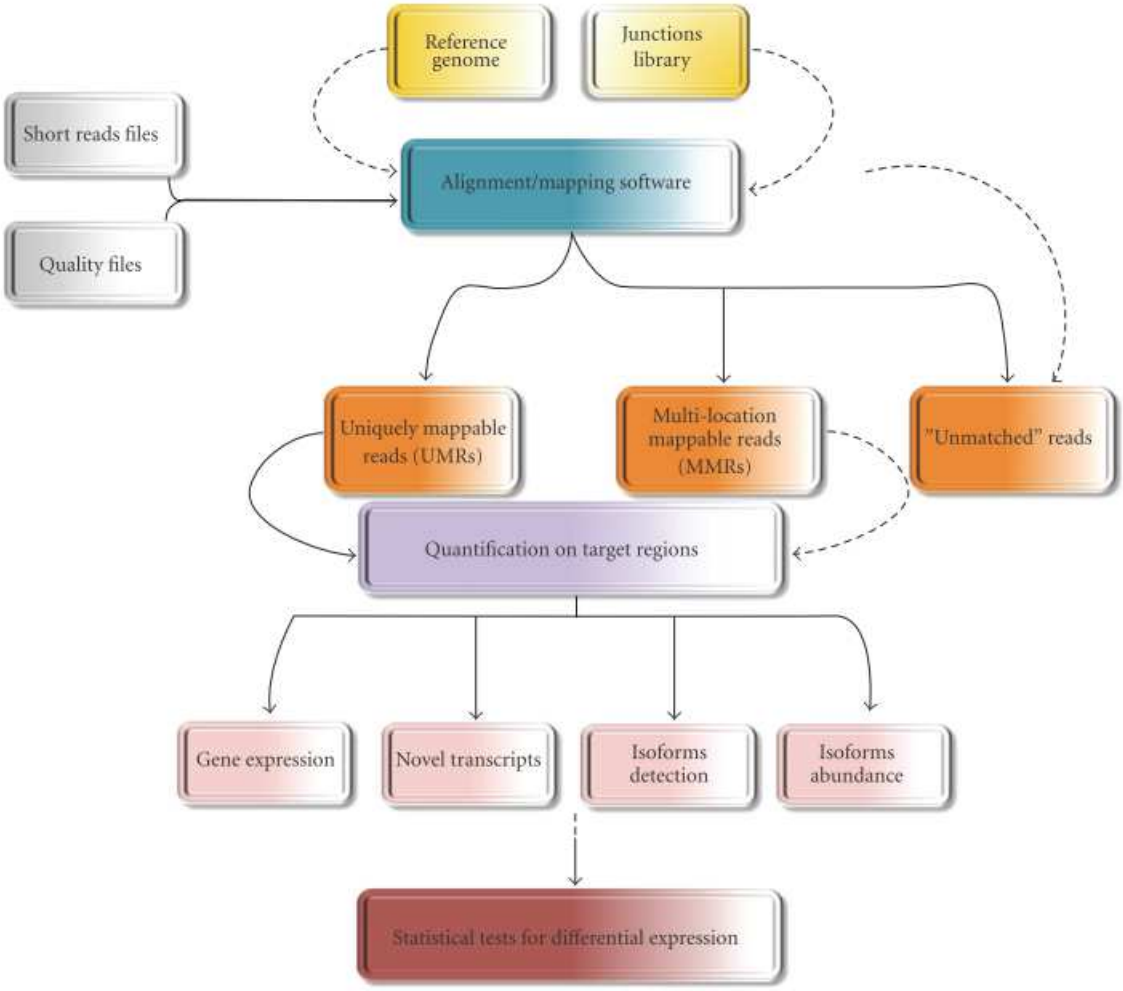


Figure 6: a generalized overview of RNA-seq analysis pipelines. Particular focus here is first demonstrated on the quality control (QC) processes that raw read files undergo prior to serving as input for read alignment packages. Three types of aligned reads emerge, with uniquely mapped reads being the most straightforward to deal with; multi-mapping and un-mapped reads each present their own set of separate bioinformatic challenges. Mapped reads can be fed into quantification analysis methods, which are capable of generating a variety of output, including identification of alternative isoforms for existing genes, novel isoforms for prospective pseudogenes, and in-sample normalized read counts. The normalized in-sample read values [or raw read counts, depending on the mathematical model employed by the subsequent analysis suite] are then used as input for differential expression analysis.

R. K. Patel and M. Jain rightly note that NGS data often suffers from a number of key sequencing artifacts in its raw short-read output; these issues include base-calling errors that stem from inaccuracies present within the sequencing platform, short insertions-or-deletions of erroneous sequence data (often referred to as indels), and contamination with primer and/or adapter sequences.

Failure to comprehensively address quality control (e.g. QC) issues concerning NGS datasets can easily impose significant consequences for downstream analysis, including read alignment, transcript quantification, and even differential gene expression analysis results between samples within a study. [27] In addressing this challenge, a multitude of NGS quality control toolkits and packages have emerged in the past decade, with select examples including the FASTX-Toolkit, FastQC, and the NGS QC Toolkit; an example output view from FastQC follows.

Figure 7: Sample Output of Read Quality from FastQC [28]

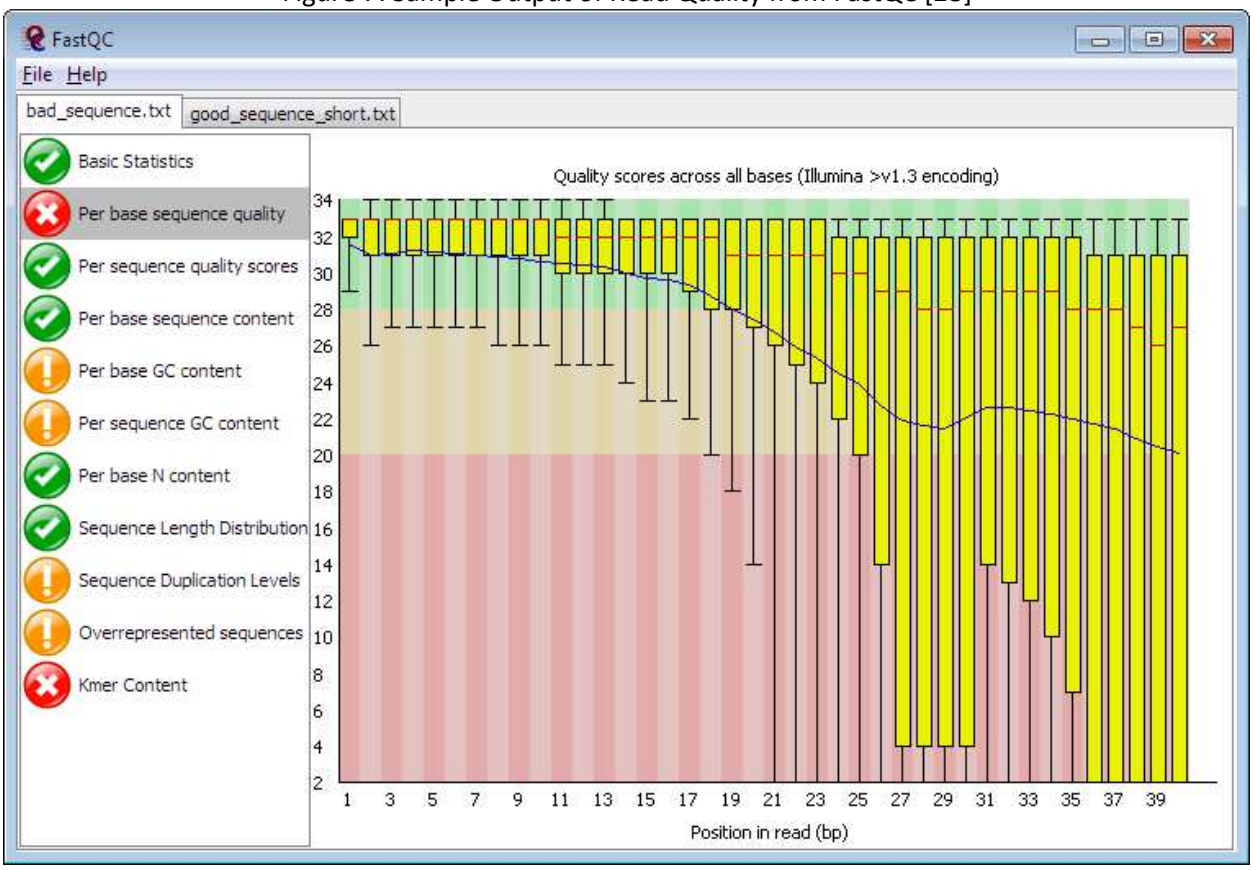


Figure 7: sample output from a run of FastQC on sequenced reads within a FASTQ dataset. Phred scores are linearly reported on the vertical axis, with a higher Phred score indicating a logarithmically higher probability that the sequencing platform has correctly called the nucleotide identity at that particular base-pair coordinate for a given read. Base-pair positions along the sample set of reads are given on the horizontal axis.

Next-generation sequencing data is often reported in the near-universal FASTQ format [or one of its variants], in which Phred quality scores accompany each nucleotide along the length of short-reads output by an NGS platform. Phred scores logarithmically indicate the probability that a sequencer has erroneously miscalled a particular nucleotide within a read and were first described by B. Ewing and P. Green at the Cold Spring Harbor Laboratory in the late 1990s [29]:

$$Q_{Phred} = -10 * \log_{10}(P(error))$$

$$P(accurate) = 1 - 10^{\left(\frac{Q_{Phred}}{-10}\right)}$$

Thus, a Phred quality score of 10 indicates that there is a 90% probability that the sequencer accurately identified the given nucleotide, a score of 20 indicates a 99% probability of accuracy, a score of 30 possesses a 99.9% probability of accuracy, and so on. The QC process in many RNA-seq studies often additionally involves clipping of adapter sequences from reads.

Following quality control of raw RNA-seq data, analysis pipelines will typically perform alignment of "cleaned" reads against a reference, which may be a known or *de novo* assembled genome or transcriptome. As noted by Oshlack *et al*, there are several categories of algorithms employed by modern read alignment tools [26], chief of which is the Burrows-Wheeler transformation, employed by the Bowtie-TopHat-Cufflinks pipeline that has been standardized into use as the "Tuxedo" pipeline employed by a great number of published RNA-seq studies covering a myriad of model and non-model organisms [30]. The first component of the Tuxedo pipeline, Bowtie, was initially presented by B. Langmead *et al* as an ultra-fast, memory efficient alignment tool capable of aligning tens of millions of short reads against a given reference sequence per CPU-hour. While employing the Burrows-Wheeler transformation to index a given reference sequence, Bowtie additionally introduces a novel set of algorithms enabling efficient back-tracking through mismatch-containing partial sequence matches found within the Burrows-Wheeler index. Bowtie thusly allows for alignment of error-containing reads and variant-prone references at low computational memory consumption. [31]

Figure 8: The Burrows-Wheeler Transformation [31]

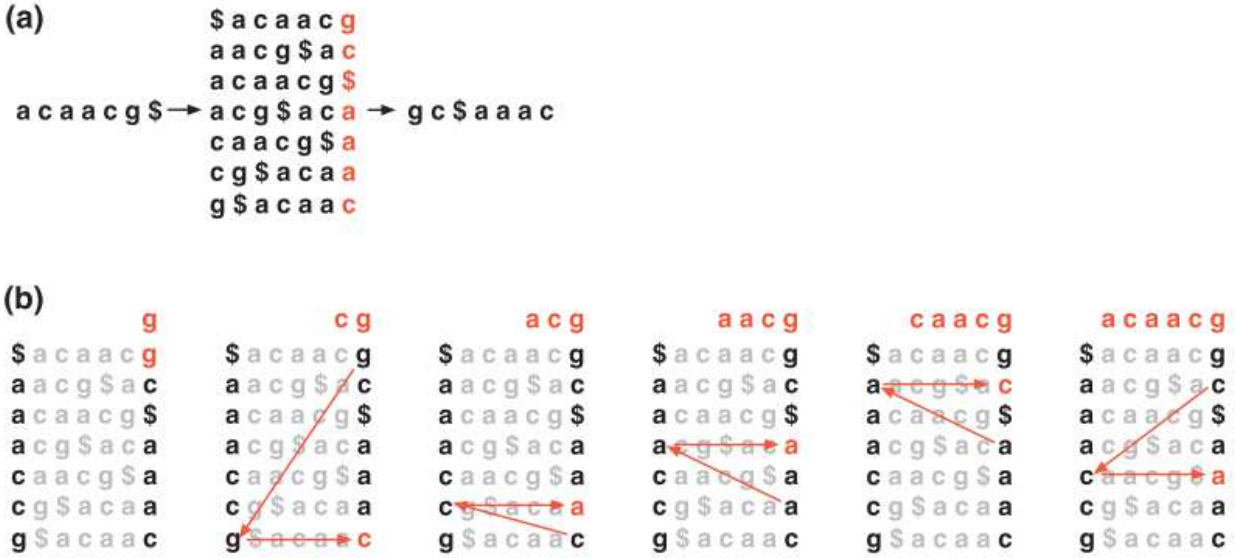


Figure 8: a sample demonstration of the Burrows-Wheeler Transformation (BWT) given by Langmead *et al*. In **a**, the BWT matrix of original sequence *T*, "acaacg", is generated—where the character "$" represents a string not in *T* and also less than the characters found within *T*. Thus, the Burrows-Wheeler transformation of *T*, BWT(*T*), is generated by forming a matrix composed of all the cyclic rotations of *T$* and re-ordering the matrix lexicographically; the right-most column corresponds to BWT(*T*), "gc$aaac", and is colored in red. In **b**, the BWT simplification lemma of "last first (LF) mapping" is demonstrated, where LF mapping denotes that the i^{th} occurrence of some character *x* in the right-most column of the BWT(*T*) matrix corresponds to the exact character mapped by the i^{th} occurrence of the same character *x* in the left-most matrix column. In this way, BWT(*T*) can be employed to regenerate the original sequence *T* and forms the indexing basis of memory-efficient read alignment in Bowtie.

In the next major step of the Tuxedo pipeline, reads aligned via Bowtie are used as input for TopHat, a splice junction-discovering tool initially published by C. Trapnell, L. Pachter, and S. Salzberg in 2009. In theory, an RNA-seq analysis could more simply be performed by directly aligning quality-controlled reads against a known reference transcriptome, yet this has often been difficult [and generally un-recommended] to perform in the past, due to whole organismal transcriptomes being absent or incomplete in the existing literature—even for well-researched model organisms. Thus, the TopHat pipeline was designed to address this issue by using whole genome alignment as a proxy for transcriptome mapping, quantification of transcript abundance, and discovery of novel gene isoforms. [32] A general workflow encompassing Bowtie and TopHat follows.

Figure 9: The Bowtie-TopHat Pipeline [32]

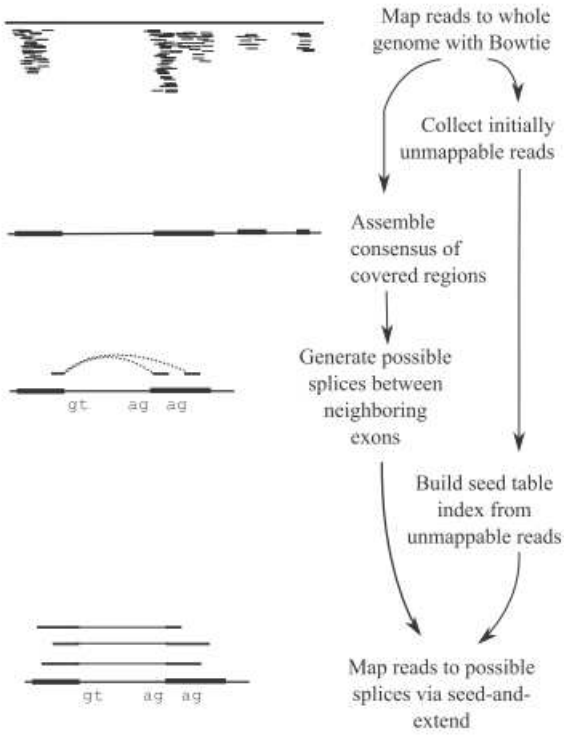


Figure 9: a generalized workflow describing the TopHat RNA-seq read alignment pipeline, beginning with input composed of reads aligned against a given reference genome via Bowtie. Output from Bowtie comprises reads grouped into one of two categories: mapped and initially un-mapped (IUM) reads. Genome-mapping reads are assembled into a consensus of covered regions, from which potential gene splicing sites are generated through identification of neighboring exons. Meanwhile, the IUM reads are assembled into a seed table index, and finally, un-mapped reads are aligned against potential splice sites via the seed-and-extend method through leveraged use of the seed table index.

Once TopHat has been used to reveal the potential set of alternatively spliced isoforms within a given transcriptome, the Cufflinks package of RNA-seq utilities—first published by C. Trapnell *et al* in 2012—may be employed to statistically quantify normalized transcript abundance at both in-sample and between-sample (e.g. differential expression) levels for an experiment. Core utilities within this bioinformatics package include: Cufflinks, an assembly generator that employs TopHat-output read

alignment files to generate a *de novo* transcriptome for each experimental condition; Cuffmerge, which merges transcriptome assemblies output by Cufflinks to provide a basis for calculating gene and transcript expression; Cuffdiff, which takes the TopHat-mapped reads and Cuffmerge-aggregated assembly to calculate statistically-tested expression levels and provides an additional layer of differential expression analysis; and CummeRbund, a visualization plotting tool that facilitates generation of volcano, scatter, and box plots for quantified RNA-seq data. [30] An illustration of the TopHat-Cufflinks pipeline follows.

Figure 10: The TopHat-Cufflinks Pipeline [30]

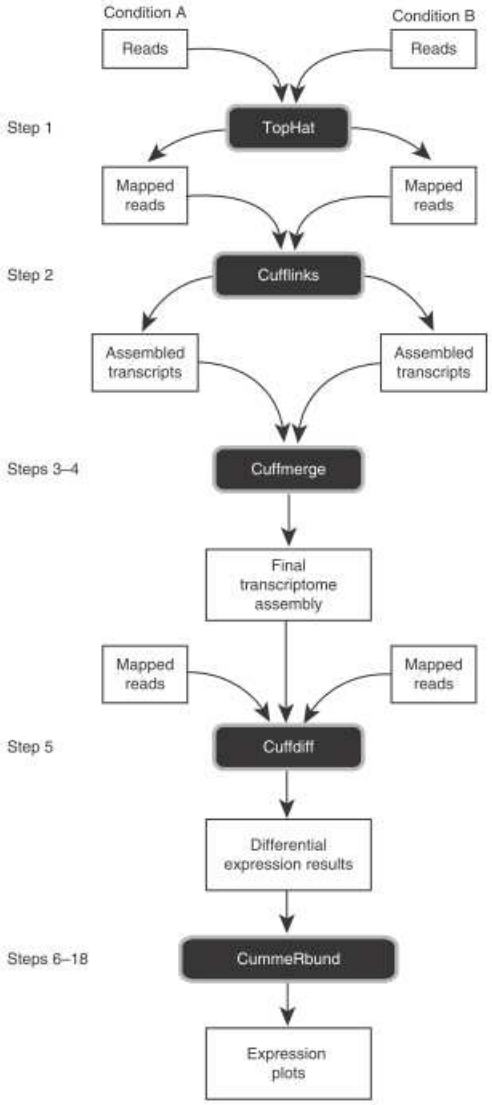


Figure 10: a depiction of the TopHat-Cufflinks pipeline for quantifying RNA-seq data. Bowtie-aligned reads are input into TopHat, which has been previously described (see Figure 9). TopHat-mapped reads are subsequently input into Cufflinks, which generates transcriptome assemblies for each experimental condition and reports in-sample transcript quantification values. Multiple assemblies are merged using Cuffmerge before being input, along with the TopHat-mapped reads, into Cuffdiff to perform statistically-tested differential expression analysis on genes or transcripts stemming from samples associated with various experiment conditions.

Gene and transcript expression values output from either the Cufflinks (e.g. in-sample normalization) or Cuffdiff (e.g. between-sample normalization, differential expression analysis) tools within the Tuxedo RNA-seq analysis pipeline typically emerge in the form of “reads per kilobase exon per million mapped reads” (RPKM) or “fragments per kilobase exon per million mapped fragments” (FPKM) units, where FPKM specifically indicates that the given RNA-seq dataset contains paired-end reads [30]. Essentially, the RPKM unit indicates that *n* many reads will generally be found per kilobase of coding DNA (cDNA) for every million mapped reads from the experiment; the unit represents a statistical normalization for the stochastic nature of read alignment observed in every NGS experiment. The FPKM unit extends this same line of thought, but with the term “fragment”, where each fragment exists in place of a set of paired-end reads. [33] The use of paired-end reads as sequencing fragments in RNA-seq originated with J. O. Korbel *et al* in mapping structural variations within the human genome; paired-end fragments facilitate higher-quality alignments in repetitive genomic regions, in identifying indels, and in presenting spaced reads that map more uniquely to a given genome or transcriptome [34].

There exists a subtle flaw in the metaphysical design behind the RPKM and FPKM units, one which L. Pachter remains vocal of: the number of mapped reads in a given sample from an experiment is virtually never constant. Thus, the unit value of one RPKM or FPKM from a given experimental sample to another, and especially from one RNA-seq experiment to another, is practically never the same. This realization suggests that these units were ill-designed, and L. Pachter *et al* have largely remonstrated their previously published [and now, unfortunately, ubiquitously employed] units into a better standardized successor, “transcripts per million” (TPM). [33, 35] These units are reported by default with use of Kallisto, a modern ultra-fast RNA-seq pseudoaligner and transcriptome quantifier published by L. Pachter *et al* [36].

The Gene Expression Omnibus and RNA-Seq Atlases

The rapid adoption and widespread use of gene expression microarrays and next-generation sequencing (NGS) platforms over the past two decades has formatively shaped the future of molecular biology and functional genomics. It comes as no surprise, then, that the voluminous nature of data produced using these methods inevitably heralded a public outcry among researchers for centralized repositories that would house published experimental data; the Gene Expression Omnibus (GEO) project at the National Center for Biotechnology Information (NCBI) was created to answer this need and was initially designed around the challenge of housing microarray experiment metadata and supplemental analysis results. [37] The schema behind the GEO database, which remains in active use today, is detailed below.

Figure 11: Schema behind the Gene Expression Omnibus [37]

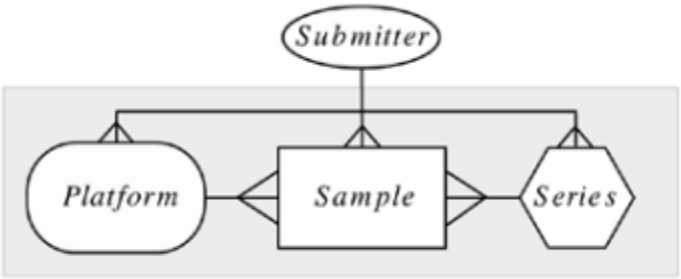


Figure 11: a depiction of the schema used in construction, maintenance, and access of the GEO database. Here, the entity-relationship diagram is specified, where each published study is referred to by a "Series" accession identifier; each series contains one or more sample accession IDs, where each sample entry in the database corresponds to a sample from the submitted experiment. Each sample may also map to *n* many platform IDs, which has historically implied the use of multiple types of microarrays in the given experiment. More recently, the presence of multiple platform accession identifiers in a given GEO series indicates use of multiple NGS platforms. Thus, the list of platforms employed in a given GEO series is found through the set union of the platforms listed through the series' constituent samples.

Despite being designed with the express objective of housing microarray study data, the Gene Expression Omnibus began accommodating user requests to store next-generation sequencing data in the late 2000s, and by 2009, the GEO team had published their initial work regarding storage mechanisms and data workflows relating to non-expression microarray data and NGS studies, which were grouped under the category of “Omics”-related data [38]. An example of user querying and viewing of study-sample metadata for a particular Rat-related series-sample entry in GEO datasets follows.

Figure 12: Example View of RNA-seq Study-Sample Metadata in GEO

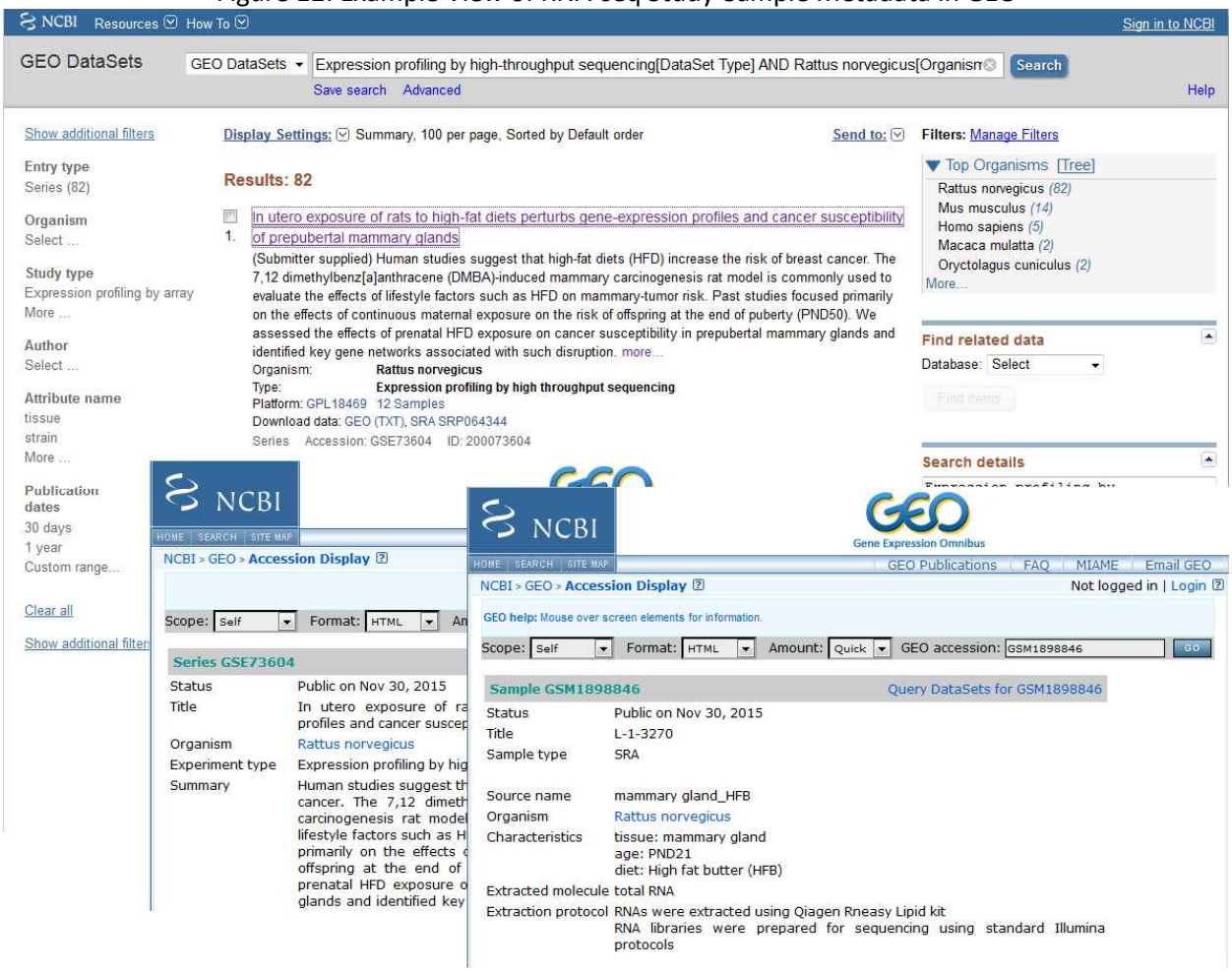


Figure 12: an example view of what GEO users see when browsing through RNA-seq studies concerning the Rat species. For each series (e.g. study) accession identifier, the user is presented with a summary page concerning the overall objectives and methodology of the experiment, including any platform accession identifiers [and their associated metadata] employed in the course of the study. These series summary pages include links to sample accession identifier pages, each of which describes the physiological conditions (including growth protocol, age, gender, strain, and many other potential parameters) under which the sample was obtained. Link-outs to study-sample raw data may be present on either the series or sample pages, and supplementary analysis (e.g. expression results) may be present on either set of pages.

Perhaps one of the greatest strengths behind the design of the Gene Expression Omnibus is its lack of stringent standards regarding user-entered data; while the database prefers to take binary alignment mapping (BAM) read files from NGS studies as the primary form of its raw data, the system also accepts a wide variety of alternative formats, including the sequence alignment mapping (e.g. SAM, uncompressed BAM) format and FASTQ (e.g. raw, un-aligned) read files. These sequence-related raw

data files are taken through GEO as an interface and are physically stored within the Short Read Archive (SRA) of NCBI, which requires use of the SRA Toolkit to extract and perform essential manipulations, such as BAM-to-FASTQ conversions on raw data [38].

Researchers are permitted to enter whatever metadata they believe to be descriptive of their experiments' underpinnings, with no absolute minimum on the types or kinds of data [or metadata] that must be entered into the GEO database. Expression analysis results within GEO datasets are also not standardized, with researchers being able to store the following equivalently: raw read counts, which represent a deeply biased perspective of actual gene/isoform expression; in-sample normalized RPKM/FPKM/TPM values, which often fall short of the true goal of many RNA-seq studies; and complete differential expression results obtained through the entirety of the previously described Tuxedo RNA-seq pipeline or equivalent differential expression (DE) workflows, including use of tools such as edgeR, RSEM, or DEseq.

The highly flexible standards of data entry for GEO give rise to a certain paradigm regarding its NGS metadata, raw data, and expression analysis results: while it is relatively straightforward for researchers to input their various types of data into the system, it is far more challenging for bioinformaticians to meaningfully extract, translate, integrate, and serve GEO-related experiment data and results to end-users in visually digestible ways. To this end, database projects designed to serve as repositories and atlases of RNA-seq data have emerged in recent years, with the European Expression Atlas serving as perhaps the most canonically well-known example. Analysis results within the Expression Atlas are stored and displayed through two fundamental models: "baseline expression" for a gene entry within a sample (e.g. in-sample normalization) and "differential expression" (e.g. between-sample normalization) computed among samples within a given study. [39]

It is critically important to note that RNA-seq results stored and displayed by the Expression Atlas are *not* the original results deposited by researchers into GEO and subsequently published into

existing scientific literature; the Expression Atlas project employs a homegrown RNA-seq pipeline as a "one-size-fits-all" model for re-computing expression results [40]. Thus, this approach technically classifies as a form of secondary re-analysis for RNA-seq datasets, and with this method, there exist both advantages and disadvantages. Positively, previous studies that lacked complete RNA-seq analyses (e.g. those that do not proceed beyond base-calling or determining read-counts per gene loci) justly receive ideally complete results for downstream ontology mapping and pathway analysis. However, the act of forcing—and exclusively presenting—secondary re-analysis, particularly for datasets whose original analyses were sufficiently complete and properly submitted to GEO, generates potentially conflicting results displayed by the Expression Atlas. Solely presenting secondary results, which may be discordant with their original counterparts, deviates from that which has been published in the literature and may undesirably misinform users.

The Research Question

One might resultantly envision a scenario in which ideally translated GEO RNA-seq results are presented adjacently beside secondary re-analysis results, such as those produced by the Expression Atlas. The work described hereafter was guided by this vision and comprises an effort to effectively and meaningfully extract, translate, and database-load GEO series and sample metadata and to subsequently link these metadata to translated [and rehabilitated, wherever necessary] supplemental expression analysis results. Thus, the primary research question behind this work: "how can one transparently and reproducibly construct an RNA-seq atlas from stored metadata and supplemental analyses within the Gene Expression Omnibus?" This research question is divided into three series of constituent questions addressed by the following chapters:

1. How can one reproducibly construct a relational database store from series and sample metadata within GEO?
2. Of what use is the metadata gathered and relationally stored from the previous step? How can sample-specific characteristic attributes be further atomized and loaded?
3. How might supplemental RNA-seq analysis results—for which no robust standard exists or is imposed upon their structure—be translated and relationally appended to the database generated from the previous steps?

In the following chapter, the design, implementation, and functionality of GEOMP—a metadata parser and relational database constructor for GEO next-generation sequencing datasets—is described. In chapter 3, a retrospective analysis of past bioinformatics methods is described, with the purpose of investigating the capacity to reproduce past biomedical informatics research; this work is fundamentally enabled by the metadata extracted, better atomized, and database-loaded by GEOMP2. In chapter 4, a prototypical implementation of GEORGET—an RNA-seq gene expression results translator for GEO-submitted NGS studies—is described, with results presented from testing of the translation model upon a zebrafish training superset and two randomly selected supersets of mouse and human RNA-seq data. Chapter 5 concludes this work, with discussion of the many paths in which this work will continue to mature in future years.

References

1) Crick F. Central Dogma of Molecular Biology. Nature. 1970; 227 (5258): 561-563.

2) Bell S P and Dutta A. DNA Replication in Eukaryotic Cells. Annual Review of Biochemistry. 2002; 71: 333-374.

3) Lee T I and Young R A. Transcription of Eukaryotic Protein-Coding Genes. Annual Review of Genetics. 2000; 34: 77-137.

4) Abbondanzieri E A, Bokinsky G, Rausch J W et al. Dynamic Binding Orientations Direct Activity of HIV Reverse Transcriptase. Nature. 2008; 453 (7192): 184-189.

5) Dalmay T, Hamilton A, Rudd S et al. An RNA-Dependent RNA Polymerase Gene in Arabidopsis is required for Posttranscriptional Gene Silencing Mediated by a Transgene but not by a Virus. Cell. 2000; 101 (5): 543-553.

6) Uzawa T, Yamagishi A, and Oshima T. Polypeptide Synthesis Directed by DNA as a Messenger in Cell-free Polypeptide Synthesis by Extreme Thermophiles Thermus thermophilus HB27 and Sulfolobus tokodaii strain 7. Journal of Biochemistry. 2002; 131 (6): 849-853.

7) Anraku Y, Mizutani R, and Satow Y. Protein Splicing: Its Discovery and Structural Insight into Novel Chemical Mechanisms. IUBMB Life. 2005; 57 (8): 563-574.

8) Starokadomskyy P L. Protein Splicing. Molecular Biology. 2007; 41 (2): 278-293.

9) Kovacs G G and Budka H. Prion Diseases: from Protein to Cell Pathology. The American Journal of Pathology. 2008; 172 (3): 555-565.

10) Kanno T, Aufsatz W, Jaligot E et al. A SNF2-like Protein Facilitates Dynamic Control of DNA Methylation. EMBO Reports. 2005; 6 (7): 649-655.

11) Bird A. DNA Methylation Patterns and Epigenetic Memory. Genes & Development. 2002; 16 (1): 6-21.

12) Sanger F, Nicklen S, and Coulson A R. DNA Sequencing with Chain-Terminating Inhibitors. Proceedings of the National Academy of Sciences. 1977; 74 (12): 5463-5467.

13) Sanger F and Coulson A R. A Rapid Method for Determining Sequences in DNA by Primed Synthesis with DNA Polymerase. Journal of Molecular Biology. 1975; 94 (3): 441-446.

14) França L T C, Carrilho E, and Kist T B L. A Review of DNA Sequencing Techniques. Quarterly Review of Biophysics. 2002; 35 (2): 169-200.

15) Anderson S. Shotgun DNA Sequencing using Cloned DNase I-generated Fragments. Nucleic Acids Research. 1981; 9 (13): 3015-3027.

16) Venter J C, Adams M D, Myers E W et al. The Sequence of the Human Genome. Science. 2001; 291 (5507): 1304-1351.

17) Lander E S, Linton L M, Birren B, and the International Human Genome Sequencing Consortium. Initial Sequencing and Analysis of the Human Genome. Nature. 2001; 409 (6822): 860-921.

18) Metzker M L. Sequencing Technologies – the Next Generation. Nature Reviews Genetics. 2010; 11 (1): 31-46.

19) Costa V, Angelini C, De Feis I, and Ciccodicola A. Uncovering the Complexity of Transcriptomes with RNA-Seq. Journal of Biomedicine and Biotechnology. 2010.

20) Wang Z, Gerstein M, and Snyder M. RNA-Seq: a Revolutionary Tool for Transcriptomics. Nature Reviews Genetics. 2009; 10 (1): 57-63.

21) Marguerat S and Bähler J. RNA-seq: from Technology to Biology. Cellular and Molecular Life Sciences. 2010; 67 (4): 569-579.

22) Ozsolak F and Milos P M. RNA Sequencing: Advances, Challenges and Opportunities. Nature Reviews Genetics. 2011; 12 (2): 87-98.

23) Van Dijk E L, Auger H, Jaszczyszyn Y, and Thermes C. Ten Years of Next-Generation Sequencing Technology. Trends in Genetics. 2014; 30 (9): 418-426.

24) Schadt E E, Turner S, and Kasarskis A. A Window into Third-Generation Sequencing. Human Molecular Genetics. 2010; 19 (2): 227-240.

25) Feng Y, Zhang Y, Ying C et al. Nanopore-based Fourth-generation DNA Sequencing Technology. Genomics, Proteomics, and Bioinformatics. 2015; 13 (1): 4-16.

26) Oshlack A, Robinson M D, and Young M D. From RNA-seq Reads to Differential Expression Results. Genome Biology. 2010; 11: 220.

27) Patel R K and Jain M. NGS QC Toolkit: A Toolkit for Quality Control of Next Generation Sequencing Data. PLoS ONE. 2012; 7 (2): e30619.

28) Andrews S. FastQC: a Quality Control Tool for High Throughput Sequence Data. Babraham Bioinformatics. 2010.

29) Ewing B and Green P. Base-Calling of Automated Sequencer Traces Using Phred – II – Error Probabilities. Genome Research. 1998; 8: 186-194.

30) Trapnell C, Roberts A, Goff L et al. Differential Gene and Transcript Expression Analysis of RNA-seq Experiments with TopHat and Cufflinks. Nature Protocols. 2012; 7 (3): 562-578.

31) Langmead B, Trapnell C, Pop M, and Salzberg S. Ultrafast and Memory-efficient Alignment of Short DNA Sequences to the Human Genome. Genome Biology. 2009; 10 (3): R25.

32) Trapnell C, Pachter L, and Salzberg S L. TopHat: Discovering Splice Junctions with RNA-seq. Bioinformatics. 2009; 25 (9): 1105-1111.

33) Pimentel H. What the FPKM? A Review of RNA-seq Expression Units. The Farrago. 2014.

34) Korbel J O, Urban A E, Affourtit J P et al. Paired-End Mapping Reveals Extensive Structural Variation in the Human Genome. Science. 2007; 318 (5849): 420-426.

35) Pachter L. Stories from the Supplement. Genome Informatics Keynote Address, Cold Spring Harbor Laboratory. 2013.

36) Bray N, Pimentel H, Melsted P, and Pachter L. Near-optimal RNA-seq Quantification. bioRxiv. 2015.

37) Edgar R, Domrachev M, and Lash A E. Gene Expression Omnibus: NCBI Gene Expression and Hybridization Array Data Repository. Nucleic Acids Research. 2002; 30 (1): 207-210.

38) Barrett T, Troup D B, Wilhite S E et al. NCBI GEO: Archive for High-throughput Genomic Data. Nucleic Acids Research. 2009; 37: D885-D890.

39) Petryszak R, Keays M, Tang Y A et al. Expression Atlas update—an integrated database of gene and protein expression in humans, animals and plants. Nucleic Acids Research. 2016; 44 (D1): D746–D752.

40) Petryszak R, Fonseca N A, Füllgrabe A et al. The RNASeq-er API—a gateway to systematically updated analysis of public RNA-seq data. Bioinformatics. 2017; 33 (14): 2218–2220.

41) Pall G S and Hamilton A J. Improved northern blot method for enhanced detection of small RNA. Nature Protocols. 2008; 3 (6): 1077-1084.

42) Heid C A, Stevens J, Livak K J, and Williams P M. Real Time Quantitative PCR. Genome Research. 1996; 6: 986-994.

43) Vandesompele J, De Preter K, Pattyn F et al. Accurate normalization of real-time quantitative RT-PCR data by geometric averaging of multiple internal control genes. Genome Biology. 2002; 3 (7): research0034.1-11.

Chapter 2: GEOMP, a Metadata Parser and Relational Database Constructor for the Gene Expression Omnibus

Abstract

The proliferation of high-throughput next-generation sequencing technologies over the past decade has given rise to a voluminous increase in the amount of data generated by modern biological and clinical studies. The Gene Expression Omnibus (GEO), developed as part of the National Center for Biotechnology Information at the National Institutes of Health, was originally designed as an international repository for housing standards-compliant microarray study metadata and expression profiles but has since been updated to handle next-generation datasets. The flexible design of the GEO database, which allows researchers to specify their own attributes and thereby modify the metadata schema of GEO itself, presents a challenge to bioinformaticians aiming to construct databases from studies housed within GEO.

We present GEOMP, a package of utilities designed to facilitate and expedite the initiation of a database project from GEO study metadata. In its first stage, GEOMP directly queries the Gene Expression Omnibus through use of the Entrez programming utilities and the GEO application program interface. In its main stage of execution, GEOMP parses through the simple omnibus format in text files pulled from GEO and performs any combination of several possible actions: output of a spreadsheet through optionally defined user attributes, construction of a SQLite database from parsed metadata, and output of unique analysis methods specific to each queried study. GEOMP is capable of automatically constructing its own configuration containing the most frequently occurring metadata attributes and may additionally be used to pull studies' publication archives from PubMed Central. GEOMP has been implemented in Bash and Modern Perl.

GEOMP aims to serve as an ease-of-use, low-dependency solution to the issue of converting a GEO search query into structured, parsed, and relational database-loaded metadata. With significant room for continued refinement and additional features, we plan to expand GEOMP in the future.

Background

In 2000, NCBI established the Gene Expression Omnibus (GEO) to serve as a public data repository for a variety of functional genomics experiments, most predominantly those related to microarray platforms at the time [1]. The infrastructure of the GEO database was designed with great flexibility in mind, such that microarray experiment data complying with the MIAME standard [3] could be inserted without significant difficulty [1, 2]. With the advent and rapid rise of next-generation sequencing (NGS) techniques over the past decade, the flexibility of the GEO database provided an advantage in allowing for accommodation of NGS dataset structures without requiring a fundamental redesign of infrastructure [1, 2, 4]. Currently, GEO houses data from over 76,000 experiments spanning over two million samples; more than 10,700 of those studies have been performed through use of NGS platforms.

Structurally, the surface of the GEO database consists of tables without limits on the number of rows or columns granted to tab-delimited input data; this characteristic confers malleability to the core database in adapting to a variety of data structures from microarray- and NGS-related studies. Certain columns from input data are recognized for special reserved meaning, and data from these columns are extracted to secondary databases that form the core of GEO, which is comprised of three types of entities: series, sample, and platform records. Series records are given the “GSE” prefix and define the set of samples contained within a given experiment, while also describing the overall research goal and methods of the experiment. A sample record is prefixed with “GSM” and contains experimental condition information and any unique analysis methods used for a particular sample within its parent

(e.g. series, GSE) study. Platform records are prefixed with “GPL” and contain general information regarding a given microarray or sequencing platform and also house references to every series and sample accession the platform has been used for; platform records thusly have a tendency to balloon quickly. [1]

Given the significant plasticity of the external GEO database tables, which are later translated into the core of the database, it stands to reason that researchers providing novel metadata attributes therefore redefine the absolute schema of GEO; this observation has been confirmed by the GEO development team [1]. Resultantly, bioinformatics tools aiming to fully interface with GEO must ensure that total metadata heterogeneity of the database is captured. Previous work by Davis and Meltzer (2007) achieved this objective in the heyday of microarrays and the infancy of NGS techniques through development of GEOquery, a utility designed to fetch and parse individually specified series and sample accession numbers [5]. Zhu and Davis *et al* (2008) employed GEOquery in constructing GEOmetadb, a database project designed to leverage parsed metadata in building an alternative search engine for the core GEO database [6].

An exciting trend has emerged in recent years regarding the construction of expression atlases and databases downstream from resources such as GEO and ArrayExpress [7]. The Expression Atlas, a large web-tool component of ArrayExpress initiated by the European Bioinformatics Institute (EBI) in 2008, presents a variety of visualization schemes for RNA-seq data secondarily reanalyzed at both the in-sample and between-sample (e.g. differential expression) levels [8]. GeneAnalytics, an embedded web module of the commercial GeneCards suite developed at the Weizmann Institute of Science, employs a proprietary reanalysis strategy for RNA-seq data derived from the GTEx consortium [9, 10]. Gundersen and Jagodnik *et al* (2016) recently published GEN3VA, a web resource enabling on-the-fly 3D principal component analysis, in addition to enrichment vector analysis across microarray expression datasets spanning multiple studies, allowing for facile meta-analysis [11]. The Lair, a recently published database

by Pimentel, Sturmfels, and Pachter *et al* (2016), performs secondary reanalysis of RNA-seq data from the Short Read Archive (SRA) through use of bleeding-edge quantification tools and visualizes the results for exploration using the R Shiny package for its web interface [12-14].

There is a growing movement in the bioinformatics community towards the construction of databases that aggregate, translate, and (re)analyze published expression data and metadata, in both microarray and next-generation sequencing formats. These projects require teams of bioinformaticians with significant pooled experience to initiate, in part due to a lack of foundational tools that interface fully with well-established resources such as GEO and ArrayExpress. It was for this reason that we elected to develop GEOMP; its purpose, and that of its successors, is to serve as a catalyst for accelerating development of future database expression and networking projects.

Implementation

Execution of the GEOMP workflow is achieved through use of three tools, whose collective domain of operations is outlined in Figure 13. In the first stage, geomp_pull is called from the command line interface (CLI) using a GEO-compatible search query; the tool pads the query with proper spacing characters before triggering an HTTPS request to the NCBI eSearch E-Utility [15]. Output from eSearch emerges in XML format, from which geomp_pull extracts relevant unique identifiers (UIDs). Each UID is then passed as another request to the eFetch E-Utility, which returns another XML file that geomp_pull stores in memory while extracting a valid GEO accession identifier. The accession identifier is passed as part of a request to the GEO URL construction application program interface (API), which returns a Simple Omnibus Format in Text (SOFT) file [1]. Finally, the deposited SOFT files are scanned for platform-to-sample mappings that are subsequently removed; these mappings often point to samples from studies outside the domain of the original GEO query or to studies employing organisms undesired by the querying user. In our experience, removal of these superfluous mappings can potentially reduce

SOFT files by up to 90% of their original volume; mappings between samples and their sequencing platforms are preserved through the data structure of sample files.

Figure 13: the operational domain of GEOMP

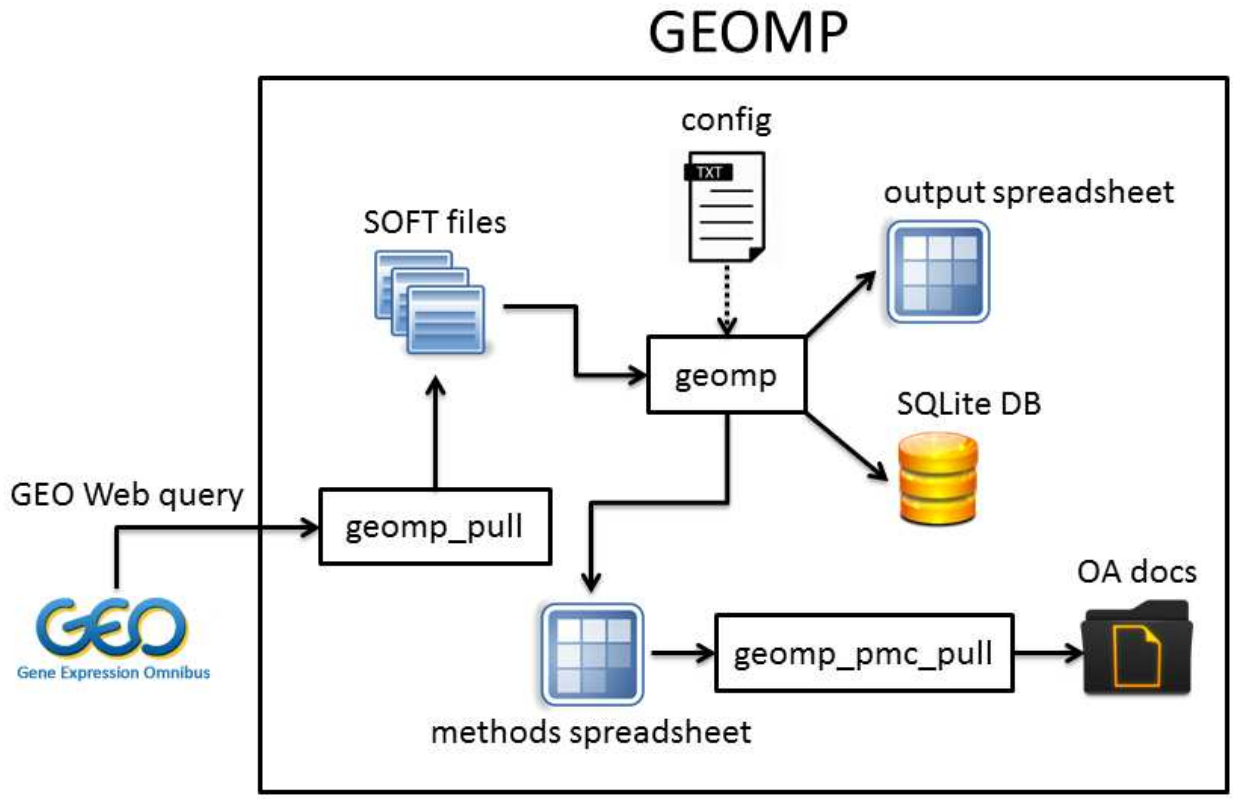


Figure 13: the GEOMP workflow begins with specification of a GEO Web query to geomp_pull, which produces a directory of SOFT metadata files. A user-defined configuration file is subsequently provided [or automatically generated] for geomp, which produces three possible outputs: a generalized metadata spreadsheet, an atomized SQLite relational database, and a spreadsheet enumerating bioinformatics methods unique to each queried study. The methods-oriented output may additionally serve as input for geomp_pmc_pull, which generates a directory tree containing Open Access publication archives.

The main stage of the workflow is performed by geomp itself, which has been implemented in functionalized Modern Perl and features both single- and double-flagged input parameters in the Unix style of tool development. At minimum, geomp requires only one flag to be specified for execution: a directory of SOFT files to interpret. Optionally, users of geomp may specify a species name to generate a study-specific metadata attribute detailing the number of child samples that match the provided

species. Additionally, a configuration file listing desired series and sample metadata attributes may be provided to geomp. A brief example of a possible configuration file:

```
[ GSE ]
geo_accession
title
pubmed_id

[ GSM ]
geo_accession
title
description
characteristics_ch1
```

If no user-defined configuration file is provided, geomp will automatically generate a configuration through the -a flag using the most frequently occurring metadata attributes; this defaults to a threshold of reaching the N95 of the provided dataset in descending order of the most frequently occurring metadata attributes. Alternatively, a user may override the -a flag with their own N-threshold or provide a percentage threshold for geomp to reach in constructing its configuration.

Standard output via the -o flag in geomp will result in a tab-delimited spreadsheet representing a relational join between every series and its constituent samples, with series and sample attribute columns taken from the provided configuration file. The geomp program is also capable of simultaneously constructing an attribute-atomized SQLite database, given this same configuration file, with one major caveat: the DBD::SQLite Perl module is required to enable this functionality and is the only known major dependency required for this workflow. Another minor caveat to consider: if one opts to provide a manually specified configuration file, then the “geo_accession” attribute should be listed first among both the GSE and GSM attribute lists, as this attribute functions ideally as a primary key. Further, geomp can generate an additional tab-delimited table displaying the bioinformatics analysis methods unique to and used within each study; this is particularly useful for conducting comparative meta-analyses of published bioinformatics work, a subject we intend to revisit in the future.

The third and final stage of the workflow allows for PubMed archives of interpreted GEO studies to be pulled from PubMed Central (PMC) via the geomp_pmc_pull utility, which leverages the PMC API and its FTP service and tools [16]. Following generation of a list of PubMed IDs (PMIDs) from parsed GEO methods metadata, geomp_pmc_pull interfaces with the PMC ID converter to cross-reference a viable PubMed Central ID (PMCID) for the given PMID. If successful, the tool then makes a request to the PMC Open Access (OA) resource to obtain a list of valid FTP links from which to disseminate archive data. If a list presents with PDF and/or tarball links, geomp_pmc_pull obtains the listed content and organizes it within directories named after the PMID of the given study. Due to the heavy continuous server load exerted upon PMC, geomp_pmc_pull sleeps for several seconds in-between each of its constituent processes.

Results and Discussion

A critical concern repeatedly revisited throughout the development process of GEOMP relates to its future role: how does it differ from its predecessor, and how can it be expanded upon? In addressing the former, we emphasize the lightweight and flexible architecture of the GEOMP workflow, which at minimum requires system libraries that are present by default on modern Unix-compliant computing platforms. GEOquery, in comparison, requires the BioConductor package within R, in addition to at least three package dependencies: XML, RCurl, and httr. Numerous dependencies and abstraction within large, complex frameworks complicate and potentially impede the installation and usage process for potential users.

Methodologically, both GEOquery and GEOMP aim to pull, parse, and interpret study and sample metadata from GEO. However, GEOquery, in its design during the microarray era preceding the rise of NGS techniques, additionally interprets GEO-standardized microarray expression tables from curated datasets. GEOMP, in contrast, focuses on aggregation of metadata from raw GEO datasets,

identification of unique bioinformatics methods, automating creation of a relational database housing atomized metadata attributes, and fetching of PMC-related study data. Though these two workflows are similar in their initial objective of acquiring GEO metadata, their subsequent aims differ significantly. Microarray expression tables interpreted by GEOquery lend well to downstream analyses using a variety of BioConductor packages; it is for this reason that we do not view GEOMP as an absolute replacement for GEOquery, and both workflows should undergo continued development to harmonize well in future work.

A particular strength of the GEOMP workflow can be found in its first step, which enables GEOMP to facilitate a direct "query-to-data(base)" approach; users can copy a GEO Web query and provide said query as a direct argument to geomp_pull. This advantage is unavailable to users of GEOquery, who must write code to iterate through a list of studies or samples in order to process data. The flat text configuration format of geomp is an additional strength, providing the ability to freely specify GEO metadata attributes of interest to parse and potentially load into a relational database. The capacity of geomp to automatically construct its own configuration provides significant flexibility to users aiming to approach the GEO metadata problem from a naïve perspective, and updating of GEOMP-processed datasets is straightforward: one need only re-run the workflow at future points in time. It should be noted that GEOMP does not yet support interpretation and translation of supplemental analysis results from NGS studies presented within GEO; this is a complex topic, with a challenging variety of file formats and data structures that will be addressed in our future work. Immediate work following the release of GEOMP will focus on studying the composition of and atomizing the sample characteristics tags defined by researchers submitting study metadata to GEO.

Summary

The GEOMP workflow presents as a low-dependency, straightforward, and flexible solution to the challenge of transforming a GEO Web query into an atomized data model accessible through various avenues of output. Where previous and current solutions to this problem require users to compose additional code and perform significant work in order to subset and access parsed metadata, GEOMP directly provides users with an interface to target desired metadata attributes through a configuration that can be specified either manually or generated naively. Future work will expand the workflow's capacity to atomize sample characteristics tags and will modularize GEOMP within a larger scope in order to translate supplemental analyses from published RNA-seq studies.

References

1) Barrett T, Troup D B, Wilhite S E et al. NCBI GEO: archive for high-throughput functional genomic data. Nucleic Acids Research. 2009; 37: D885-D890.

2) Barrett T, Wilhite S E, Ledoux P et al. NCBI GEO: archive for functional genomics data sets—update. Nucleic Acids Research. 2013; 41: D991-D995.

3) Brazma A, Hingamp P, Quackenbush J et al. Minimum information about a microarray experiment (MIAME)—toward standards for microarray data. Nature Genetics. 2001; 29: 365-371.

4) Shumway M, Cochrane G, Sugawara H. Archiving next generation sequencing data. Nucleic Acids Research. 2010; 38: D870-D871.

5) Davis S, Meltzer P S. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. Bioinformatics. 2007; 23: 1846-1847.

6) Zhu Y, Davis S, Stephens R, Meltzer P S, Chen Y. GEOmetadb: powerful alternative search engine for the Gene Expression Omnibus. Bioinformatics. 2008; 24: 2798-2800.

7) Kolesnikov N, Hastings E, Keays M et al. ArrayExpress update—simplifying data submissions. Nucleic Acids Research. 2015; 43: D1113-D1116.

8) Petryszak R, Burdett T, Fiorelli B et al. Expression Atlas update—a database of gene and transcript expression from microarray- and sequencing-based functional genomics experiments. Nucleic Acids Research. 2014; 42: D926-D932.

9) Fuchs S B-A, Lieder I, Stelzer G et al. GeneAnalytics: An Integrative Gene Set Analysis Tool for Next Generation Sequencing, RNAseq and Microarray Data. OMICS: A Journal of Integrative Biology. 2016; 20: 139-150.

10) The GTEx Consortium. The Genotype-Tissue Expression (GTEx) pilot analysis: Multitissue gene regulation in humans. Science. 2015; 348: 648-660.

11) Gundersen G W, Jagodnik K M, Woodland H et al. GEN3VA: aggregation and analysis of gene expression signatures from related studies. BMC Bioinformatics. 2016; 17: 461.

12) Pimentel H, Sturmfels P, Bray N, Melsted P, Pachter L. The Lair: a resource for exploratory analysis of published RNA-Seq data. BMC Bioinformatics. 2016; 17: 490.

13) Bray N L, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology. 2016; 34: 525-527.

14) Pimentel H J, Bray N, Puente S, Melsted P, Pachter L. Differential analysis of RNA-Seq incorporating quantification uncertainty. bioRxiv. 2016. http://biorxiv.org/content/early/2016/06/10/058164. Accessed 9 December 2016.

15) Sayers E. Entrez Programming Utilities Help. https://www.ncbi.nlm.nih.gov/books/NBK25501/. Accessed 9 December 2016.

16) Developer Resources. https://www.ncbi.nlm.nih.gov/pmc/tools/developers/. Accessed 11 December 2016.

Chapter 3: Assessing Bioinformatics Methods Reproducibility with GEOMP2

Abstract

Scientific work possesses two essential characteristics: innovation and reproducibility. The latter of these two is a hallmark trait that distinguishes a meaningful result from being a glorified fluke. Rapid proliferation of next-generation sequencing platforms over the past 15 years has resulted in an explosive abundance of data that has been processed and analyzed in a myriad of ways. Without a well-established and enforced set of mechanisms for guaranteeing reproducibility of genomics results, leadership within the scientific community has expressed serious concerns regarding the potential presence of an apparent epidemic of irreproducible results. To investigate this critical concern, we have built upon our previous work with GEOMP—a metadata parser for the Gene Expression Omnibus—in order to inspect bioinformatics methods reproducibility across RNA-seq experiments spanning the zebrafish, mouse, and human research communities.

From our previous work with GEOMP, and in parallel with development of GEORGET—an RNA-seq results translator for the Gene Expression Omnibus—this work assessed the unique bioinformatics methods presented by 129 zebrafish, 76 mouse, and 85 human RNA-seq studies from 2010 through 2017. A five-point reproducibility rubric was generated and used to evaluate the methods, with an overwhelming majority of RNA-seq analysis workflows found to be not generally reproducible. Zero studies reported hardware specifications. Critically, the largest research communities—mouse and human—did not report their methods more reproducibly than the zebrafish community. We also present GEOMP2, an expansion upon our previous work that allows for the atomization and relational storing of characteristics tags for sample metadata pulled from the Gene Expression Omnibus.

Reproducibility concerns voiced by scientific leadership have been well-placed: we have found that a landslide majority RNA-seq analyses cannot be generally replicated. This is a crucial issue for future genomics research, and we present several partial solution avenues to explore.

Background

Scientific work is typically described by two qualities: innovation and reproducibility. The latter of these two is a hallmark trait that delineates a meaningful scientific result from being a glorified anecdote or fluke. The importance of reproducibility is so paramount, in biomedical research, that it surpasses the aim for clinical significance; results must minimally be reproducible [1]. In recent years, executive leadership at the National Institutes of Health (NIH) have expressed growing concerns that biomedical research may not be entirely reproducible, with researchers placing increasing importance on the trendiness of journals where novel research is published, rather than the fundamental capacity to replicate results. Leadership has also noted that scientists do not fully disclose, in transparency, the key details of their methods in an effort to retain a competitive advantage for publishing future work; this practice is harmful to the scientific community [2].

The past decade has borne witness to an explosion of genomic datasets with the prolific rise of next-generation sequencing (NGS) platforms, resulting in major paradigm shifts for research in numerous fields, including molecular biology, clinical genetics, and metabolic biochemistry [3-4]. Currently available second-, third-, and fourth-generation sequencers are capable of producing massively parallelized sequence reads that have overwhelmingly eclipsed first-generation Sanger sequencers in terms of per-run genomic yield, per-base accuracy, and individual read lengths [5]; this has cost-effectively enabled the realization of DNA and RNA sequencing experiments previously only imagined. RNA-seq, in particular, is a specialized sub-domain within bioinformatics centered on profiling gene expression across whole organismal transcriptomes and is key to modern research in

developmental biology, pathology, and toxicology [6]. Execution of RNA-seq pipelines is a non-trivial and involved process, minimally requiring quality control (QC) of raw sequence data, splice-aware alignment to a reference genome or transcriptome, and in-sample-normalized gene expression quantification [7]. An additional computational step is required to normalize between samples in order to obtain differential gene expression results across whole experiment datasets; the necessity of this step has sometimes been misunderstood by RNA-seq researchers [7-8].

The National Center for Biomedical Information's (NCBI) Gene Expression Omnibus (GEO) database was founded in the early 2000s and was originally designed to flexibly capture and provide for retrieval of microarray datasets [9]. With the meteoric rise of NGS platforms and both DNA- and RNA-seq workflows over the past decade, GEO has retrofitted its database structure to accommodate next-generation sequencing datasets [10]. A key aspect of GEO's database design is its highly malleable metadata layer, in which researchers are invited to define metadata attributes as they wish and may fill these self-defined attributes with whatever values deemed appropriate. From an information organization perspective, this is a formula for electronic entropy. GEO also presents matrix-formatted results for microarray experiments, whereas for RNA-seq studies, no apparent enforcement mechanism has been made to ensure that submitted results conform to the MINSEQE standard (e.g. Minimum Information regarding a Sequencing Experiment) [11]. This is an involved topic that we explore further in our work with GEORGET.

This work is primarily concerned with investigating the reproducibility of bioinformatics methods within NGS studies—specifically, RNA-seq analyses—submitted to the Gene Expression Omnibus. Given the malleable nature of GEO's metadata database layer, and how researchers are encouraged to submit whatever they wish to specify, one might ideally hope that researchers are maximally specific regarding methods needed to replicate past *in silica* methods. This optimism has been sobered by discouraging reproducibility results reported in the biological sciences [12]; if

disciplined biology “at the bench” cannot reliably report reagents used, then what hope can biomedical informatics work possess? Previous work with nested meta-analyses in medical informatics has also reported discouraging findings regarding the capacity to replicate previous clinical meta-analyses [13].

Methods

The purpose of this study was to investigate the reproducibility of bioinformatics methods employed in RNA-seq studies whose metadata and supplemental analyses have been submitted to the Gene Expression Omnibus. To facilitate this work, our previously presented GEO metadata pipeline, GEOMP, was used to pull NCBI universal IDs (uIDs) for zebrafish, mouse, and human RNA-seq studies. Given that the bodies of work behind the mouse and human research communities are more than an order of magnitude larger than that of the zebrafish community, the number of mouse and human RNA-seq studies—which numbered in the thousands—needed to be randomly reduced to the scale of the zebrafish RNA-seq superset for manual review; this was on the order of approximately 100 studies per species. GEOMP was then used to pull, translate, and load these supersets’ metadata into species-specific SQLite databases. Throughout this process, each superset’s uniquely reported bioinformatics methods were output to methods-formatted spreadsheets enabled by the ‘-m’ flag within GEOMP.

Each spreadsheet containing RNA-seq methods metadata was then imported into a single Excel workbook. A five-point recursive reproducibility scoring rubric was designed, with successively higher values being assigned to methods that minimally satisfy the requirements of lower values. As an example, an assigned reproducibility value of 3 implies that the given method has already satisfied the requirements for values 1 and 2. Rubric values correspond to the following:

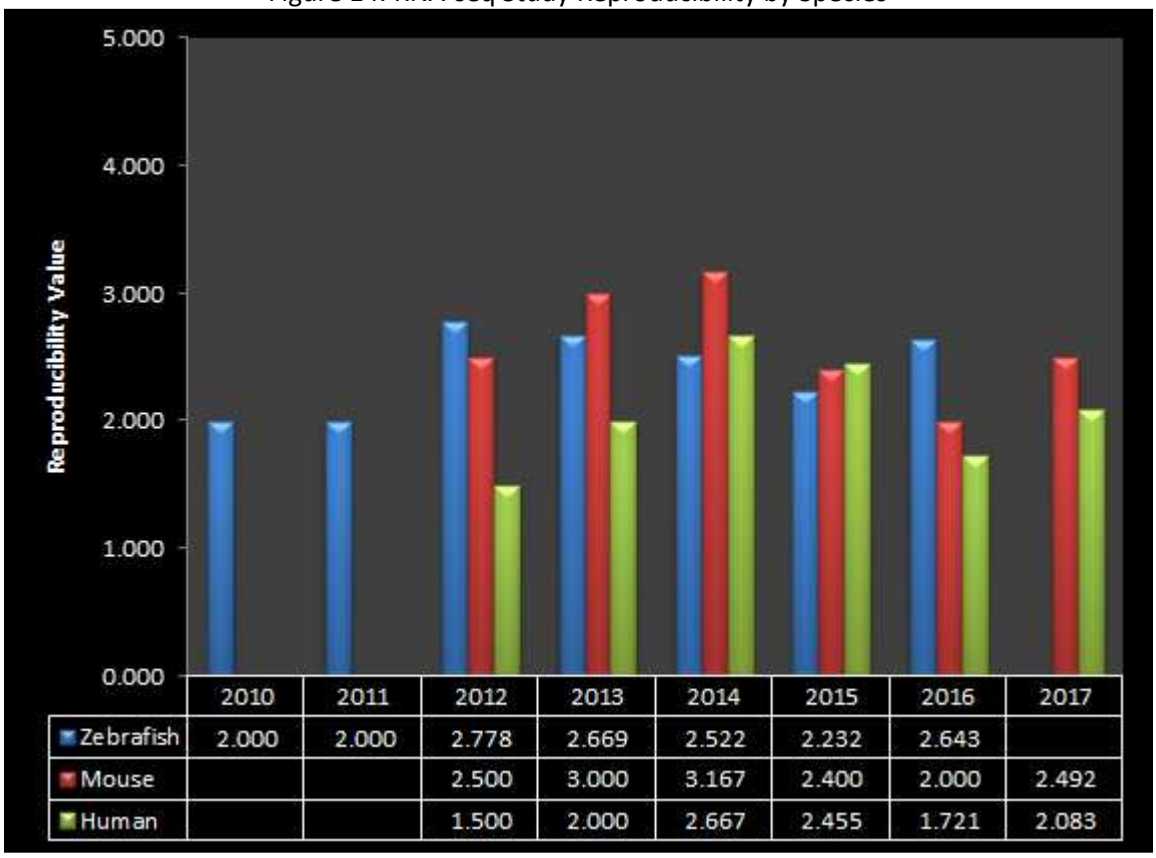
1. A bioinformatics workflow or pipeline for analysis is clearly articulated
2. Tools used in the analysis are clearly identified
3. Tool versioning is explicitly provided
4. Usage flags and/or parameters are specified for each tool in the workflow or pipeline
5. Runtime hardware specifications are well-defined

An assigned value of 4 from the rubric indicates a general capacity to reproduce previous results; a value of 5 suggests a nearly absolute ability to replicate previous results. A special note was made for methods involving the use of custom-developed tools: if no documentation regarding tool retrievability was made apparent, the given method's reproducibility score was penalized. Critically, methods with oscillating reproducibility values from step to step within their analysis descriptions were assigned an overall reproducibility score equivalent to their lowest-scoring segment. Once scores had been assigned to all reported methods, the uniquely reported methods were assigned weights based upon their frequencies of occurrence within their parent studies. The studies were then grouped by organism and year and were subsequently averaged by the number of unique studies submitted to GEO per year; these finalized values were tabulated by species and plotted.

Results

In total, bioinformatics methods reported by 129 zebrafish, 76 mouse, and 85 human RNA-seq studies were assessed for their reproducibility. Years in which a given species' RNA-seq superset had less than 4 studies were discarded from tabulation and subsequent plotting; thus, the range of years for the zebrafish superset extended between 2010 and 2016, while the range for the mouse and human supersets extended between 2012 and 2017. The average weighted reproducibility scores across all years for the zebrafish, mouse, and human supersets were found to be 2.406, 2.593, and 2.071, respectively. Figure 14 provides a multi-series histogram of the reproducibility results by year and species.

Figure 14: RNA-seq Study Reproducibility by Species



	2010	2011	2012	2013	2014	2015	2016	2017
Zebrafish	2.000	2.000	2.778	2.669	2.522	2.232	2.643	
Mouse			2.500	3.000	3.167	2.400	2.000	2.492
Human			1.500	2.000	2.667	2.455	1.721	2.083

Figure 14: RNA-seq study reproducibility by species. Reproducibility values were assigned on a five-point scale, with values above three indicating a minimally general capacity to replicate past RNA-seq studies' results. The matrix along the x-axis displays average weighted reproducibility values by species per year.

Given that a rubric-assigned reproducibility value of 4 establishes that a method is "generally reproducible", we observed that the weighted averages of studies across all organisms and inspected years fell below this minimum threshold of viability. We additionally noted that zero studies reported runtime hardware specifications for computationally demanding pipelines such as Cufflinks [8]. A recurring theme encountered when surveying studies' unique analysis methods: researchers exhibited a tendency to swap the exclusive presentation of either rubric requirement #3 or #4; in these cases, where tool usage flags were presented without explicit tool versioning, a reproducibility score of 3 was assigned. A deeply concerning observation to note from Figure 14: the average weighted reproducibility

of human RNA-seq studies falls significantly below that of either the zebrafish and/or mouse communities in 80% of the years in which all three communities were jointly assessed.

Many human study methods were assigned a reproducibility value of one, largely due to the frequent presentation of the highly ambiguous phrase, “adapters were trimmed”. What, exactly, does this mean? An Illumina sequencing reaction typically possesses four sequencing components that may be described as an “adapter” by downstream bioinformaticians: the sequence reaction’s anchoring adapter, the primer used to initiate the sequencing reaction, the sequence indices / barcodes used to multiplex samples within a run, and the upstream PCR primer; the lattermost of these is most commonly handled as the bioinformatics adapter, yet studies frustratingly lack clarity on what the “adapter” precisely is. Worse yet, the phrase “adapters were trimmed” provides no information on what tool was used to perform adapter trimming, its versioning, and whatever flags were needed to accomplish this.

Aside from the reproducibility results reported here, we also present development of GEOMP2, an upgrade to our previous workflow designed to flexibly and automatically extract, translate, and load metadata from GEO into a user-specified relational database. GEOMP2 significantly adds the capacity to atomize, aggregate, and database-load sample characteristic tags; these are sample-specific characteristic attributes that researchers supply when submitting study metadata to GEO. Since no explicit limits are made upon what researchers may supply as tags, the potential heterogeneity within these fields is vast. Users of GEOMP2 may supply a list of characteristic tags to parse and load; alternatively, users may specify a percentage- or N-based threshold of occurrence frequency for tags to be considered for normalization and database loading. The capacity to atomize, aggregate, and load sample characteristics tags is a feature absent from GEOquery, an R-based predecessor of GEOMP published more than a decade ago [14].

Discussion

Results from this study's inspection of past bioinformatics methods employed in RNA-seq studies submitted to GEO are deeply concerning: no studies reported any specification of computing hardware employed during analyses, and work performed across all research communities was found to not meet minimal requirements for being generally reproducible. Perhaps most concerning is the apparent under-performance, in terms of reproducible documentation, of bioinformatics work performed in the human research community; this is the domain that would presumably be most critical to clinically-related future applications. These results suggest that the larger bioinformatics research community continues to suffer from a systemic failure to properly mandate and enforce mechanisms for ensuring that novel research may be verified through replication. Without this assurance, it is difficult to imagine how submitted research may truly be regarded as quality scientific work.

The issue of addressing scientific reproducibility in biomedical research is regarded as a foundational challenge, described as such by researchers beginning their careers and by top leadership at the NIH [1, 2]. Executive NIH leadership has explicitly stated that the organization cannot solve this problem with the "brute force" methodology of directly funding reproducibility efforts en masse; instead, the organization has begun allocating resources towards the training of biomedical postdoctoral fellows who will strive to be more conscious of reproducibility needs in future work [2]. Others have advocated for greater incentivizing of reproducible research from journals; as an example, a publication that has been verified by replication of its results may be given special featuring by its publishing journal [1]. We echo these ideas and emphasize that the biomedical research community must rally in addressing foundational concerns regarding reproducible research; crucially, the training of future generations of researchers must place great emphasis on the need for reproducibility in order to ensure that this problem abates.

Summary

This work concerned the investigation of assessing reproducibility among bioinformatics methods submitted to the Gene Expression Omnibus by RNA-seq studies spanning the zebrafish, mouse, and human research communities. We found that the overwhelming majority of studies do not reproducibly document their analytical workflows or pipelines, further underscoring a continuing trend of troubling findings in the biomedical research domain regarding an apparent inability to replicate past studies' results. Future work along this avenue of inquiry should consider investigating methods reported directly in the literature, inspection of a larger number of studies' methods across a broader group of model research organisms, inclusion of a greater number of methods reviewers, and determination of Cohen's kappa coefficient between methods reviewers.

References

1) Russell J F. If a job is worth doing, it is worth doing twice. Nature. 2013; 496 (7443): 7-8.

2) Collins F S and Tabak L A. NIH plans to enhance reproducibility. Nature. 2014; 505 (7485): 612.

3) Schuster S C. Next-generation sequencing transforms today's biology. Nature methods. 2007; 5 (1): 16.

4) Mardis E R. The impact of next-generation sequencing technology on genetics. Trends in genetics. 2008; 24 (3): 133-141.

5) Quail M A, Smith M, Coupland P et al. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC genomics. 2012; 13 (1): 341.

6) Wang Z, Gerstein M, and Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nature reviews genetics. 2009; 10 (1): 57.

7) Oshlack A, Robinson M D, and Young M D. From RNA-seq reads to differential expression results. Genome biology. 2010; 11 (12): 220.

8) Trapnell C, Roberts A, Goff L et al. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nature protocols. 2012; 7 (3): 562.

9) Edgar R, Domrachev M, and Lash A E. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Research. 2002; 30 (1): 207-210.

10) Barrett T, Wilhite S E, Ledoux P et al. NCBI GEO: archive for functional genomics data sets—update. Nucleic Acids Research. 2013; 41: D991-D995.

11) Brazma A. Minimum information about a microarray experiment (MIAME)–successes, failures, challenges. The Scientific World Journal. 2009; 9: 420-423.

12) Vasilevsky N A, Brush M H, Paddock H et al. On the reproducibility of science: unique identification of research resources in the biomedical literature. PeerJ. 2013; 1: e148.

13) Patrick T B, Demiris G, Folk L C et al. Evidence-based retrieval in evidence-based medicine. Journal of the Medical Library Association. 2004; 92 (2): 196-199.

14) Davis S and Meltzer P S. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. Bioinformatics. 2007; 23 (14): 1846-1847.

Chapter 4: GEORGET, an RNA-seq Gene Expression Results Translator for the Gene Expression Omnibus

Abstract

The meteoric rise of next-generation sequencing technologies over the past decade and their related applications, particularly RNA sequencing analysis tools and workflows for whole transcriptome profiling, have resulted in an overwhelming abundance of scientific data and subsequently generated results. These gene expression results have canonically been stored within the Gene Expression Omnibus and have more recently been re-analyzed for visualization by the Expression Atlas. Various additional Web-based systems have emerged in recent years to allow for visualization of specific RNA-seq datasets or selected supersets, yet no known workflow apparently exists for translating canonical gene expression results stored within GEO—those which are later published into the scientific literature—into relational database-stored results which can be easily navigated for future Web application and bioinformatics tool development.

We present our initial implementation of GEORGET, a workflow comprised of numerous tools designed to function subsequent to our prior art, GEOMP, in order to pull RNA-seq results from the Gene Expression Omnibus and rehabilitate these results, where necessary, for translation and subsequent loading into a SQLite relational database that is thereafter indexed for rapid querying of translated results. The effective translation rate of GEORGET has been measured to exceed 90% among its training superset of zebrafish RNA-seq results and two randomly selected supersets of mouse and human RNA-seq results. GEORGET has been implemented with Bourne shell (e.g. Bash) in conjunction with standard Unix utilities, GNU Parallel, Modern Perl, and SQLite.

GEORGET aims to serve as the first publicly available workflow of its kind for translating RNA-seq results stored within the Gene Expression Omnibus. In contrast to other approaches, GEORGET translates deposited results as they are, such that they do not require immediate secondary re-analysis and do not deviate from what is reported in the literature. GEORGET-generated databases may ideally serve as a relational foundation for future Web applications and tools to browse next-generation sequencing datasets.

Background

The rapid proliferation of next-generation sequencing (NGS) platforms over the past decade has overwhelmingly captured the decades-long inertia previously held by first-generation Sanger sequencers [1]. This upset has been primarily owed to NGS platforms' capacity to provide genomic yield on the order of many megabases to gigabases per sequencing run, representing a per-nucleotide cost reduction spanning many orders of magnitude when compared with traditional Sanger platforms [2]. Initial second-generation platforms, such as those produced by Roche/454, Applied Biosciences (AB), and Illumina relied upon a variety of molecular and biochemical techniques in order to produce short sequencing reads at high genomic volume; these methods include pyrosequencing, bridge amplification, emulsion PCR, and DNA ligation [2-4]. When first introduced, second-generation platforms boasted sequence reads of less than 50 nucleotides (e.g. bases, or bp) in length; the length of these short reads have increased over time, such that Illumina now offers 250 bp and 300 bp paired-end kits for its modern platforms [4-6]. Third- and fourth-generation sequencers have additionally emerged within the past decade; Pacific Biosciences' RS and Sequel platforms generate circular polymerase reads (e.g. zero-mode waveguides, or ZMWs) with insert lengths on the order of ten kilobases, while Oxford Nanopore's voltage-gated platforms promise to deliver reads up to an entire megabase in length [7].

A particular focal area of NGS-enabled genomics is the domain of research known as RNA sequencing (e.g. RNA-seq), which aims to empower researchers with the capacity to profile whole organismal transcriptomes for use in both scientific and clinical applications [5, 8-9]. Execution of modern RNA-seq pipelines is a non-trivial affair, where analysis typically begins with splice-aware alignment of sequenced RNA data to a reference genome or transcriptome [10]. Examples of aligners capable of performing this step include TopHat, GSNAP, and STAR [11-13]. Following alignment, one may opt to perform full-suite RNA-seq quantification and statistical determination of differential expression between samples using a well-established pipeline; the Cufflinks package has been a canonical solution for performing this [14]. Alternatively, genomic histograms of read pile-ups may be generated from RNA-seq alignments through use of packages such as BEDTools or HTSeq [15-16]; output from these pipelines may be used as input for differential expression packages, such as RSEM, edgeR, and DESeq [17-19]. A critical aspect of RNA-seq to remain cognizant of: two stages of normalization and statistical tests are required to complete analysis—that within each sample, and a second normalization step between samples. The former accounts for the dynamic length of genes or isoforms within each sample (e.g. not every gene is of identical genomic length), while the latter—more commonly referred to as differential expression—accounts for the dynamic number of reads aligned per sample [10, 14]. Recent advances have catapulted RNA-seq analysis speed by up to two orders of magnitude: through use of pseudo-alignment algorithms, packages such as Kallisto and Sailfish now deliver accurate RNA-seq transcript quantification without incurring the heavyweight computational cost of using formal splice-aware aligners [20-21].

Given the complexity of RNA-seq analysis workflows and their myriad of interchangeable tools and resulting pipelines, it come as little surprise that a variety of Web-based frameworks have been designed to facilitate visualization of RNA-seq results. The GeneCards platform, a comprehensive [and even exhaustive] network of regulatory, pathway, and disease information for the human genome,

incorporates RNA-seq data from the GTEx consortium for exploration of gene expression by tissue [22]. The Lair has been a more recent entry from the group that designed and introduced the canonical Cufflinks RNA-seq workflow [14, 23]. RNA-seq datasets within the Lair are re-analyzed from their raw sequence data through use of Kallisto and Sleuth prior to display within an R-Shiny-enabled environment, which allows users to explore statistical trends within their datasets [20, 23-24]. The European Expression Atlas is a well-known "gold standard" in the realm of RNA-seq Web visualization platforms, combining both microarray and RNA-seq experiments collected, manually curated, and secondarily re-analyzed for well over a decade [25]. Users of the Expression Atlas can browse experiments across a wide variety of model organisms, with baseline and differential gene expression results presented with interactive plots and heat-maps; RNA-seq results within this atlas are uniformly re-analyzed with a pipeline combining TopHat2, HTSeq, and DESeq [11, 16, 19, 25].

A recurring issue with these platforms is the need to perform secondary re-analysis of RNA-seq datasets; this is particularly true with respect to both the Lair and the Expression Atlas [23, 25]. Because these analyses can be completed with a variety of methods [10], a single universal workflow does not exist for computing differential gene [or isoform] expression between samples, and so these platforms attempt to standardize analyses between studies with self-devised pipelines. Consequently, discordance exists between RNA-seq results researchers submit for publication in scientific literature and those generated from secondary re-analysis that are displayed within existing expression atlases. Thus, there exists a need to translate and database-load existing RNA-seq results from the literature, those that are stored within the Gene Expression Omnibus (GEO) [26], such that future RNA-seq atlases may be able to additionally reflect researchers' original work.

Implementation

Use of GEORGET is predicated on the existence of a SQLite database generated as output from our prior work describing GEOMP. A visual outline of the GEORGET workflow follows:

Figure 15: the GEORGET workflow

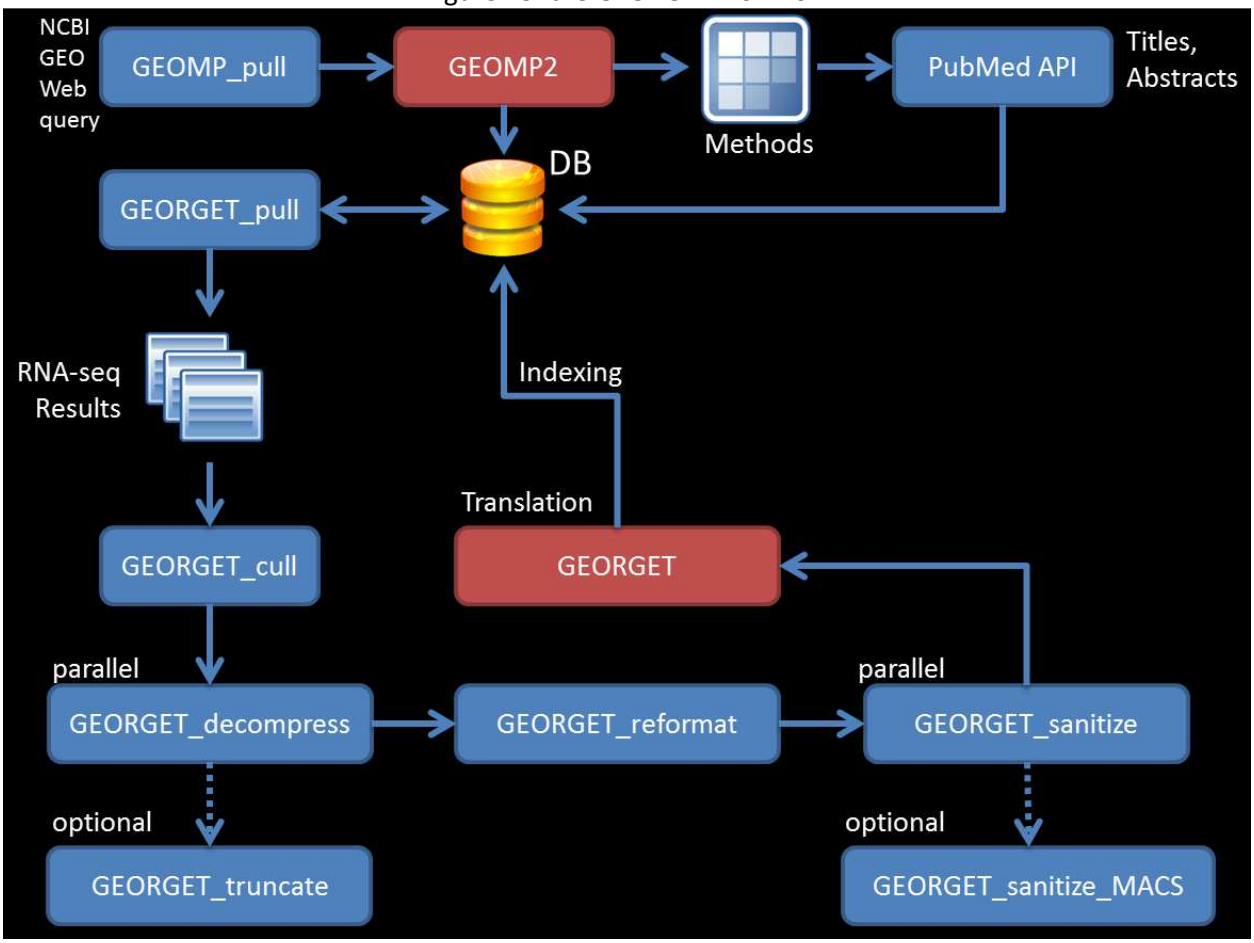


Figure 15: execution of the GEORGET workflow follows database output from GEOMP. Briefly, GEORGET_pull is used to fetch RNA-seq supplementary analysis files from the FTP store of NCBI GEO; GEORGET_decompress is then used to inflate the files' contents, optionally in parallel. GEORGET_reformat then corrects files whose line terminators are exclusively composed of carriage return characters, and GEORGET_sanitize truncates leading and terminating whitespace characters from every file. The final step, GEORGET itself, is responsible for translating, loading, and indexing the RNA-seq results into the target SQLite database.

The first step of this workflow, GEORGET_pull, begins by reading the SQLite database generated by GEOMP; the tables containing supplementary analysis files for GEO series and GEO samples (e.g. GSE

and GSM content, respectively) are read and filtered for FTP links that are not related to the Short Read Archive (SRA) and which are also not archived tarballs containing raw RNA-seq datasets. The remaining files are filtered by a size threshold, currently 250 megabytes by default, and are subsequently pulled. The following step, GEORGET_cull, is an optional one that requires an input blacklist of studies that are to be removed from the translation effort; this is needed for certain studies that are classified as RNA-seq within GEO's search querying system but do not contain canonically recognizable results. In our experience with the zebrafish training superset of GEO RNA-seq results, at minimum one study employed gene editing techniques to modify an organism's transcriptome without performing gene expression quantification or differential expression, thereby making its results unusable in the context of this work.

Results pulled from the GEO FTP store arrive in gzip format; GEORGET_decompress allows one to inflate these files' contents, with GEORGET_decompress_parallel expediting this process through use of GNU Parallel [27]. One may then optionally use GEORGET_truncate, which scans the inflated analysis result files for those that pass a defined threshold in file size; those that pass the threshold are removed from the translation process. This step may be skipped by default, as GEORGET also contains a threshold for ignoring files that surpass a given threshold in file size. GEORGET_reformat is then called, which scans all files in a given superset directory for those that are exclusively line-terminated with the carriage return character; these files are then converted to Unix-compatible format. The final mandatory pre-translation step, GEORGET_sanitize, has both a serial and parallel implementation available; this program removes the beginning and trailing whitespace characters from every RNA-seq analysis result file in the specified superset directory. Optionally, GEORGET_sanitize_MACS may also be run immediately prior to translation; this program prepares output from the model-based analysis for ChIP-seq (MACS) package for database loading [28].

GEORGET itself, the final step of this workflow, begins by connecting to the target SQLite database and verifying the existence of RNA-seq supplementary analysis tables for both GSE and GSM entries. The existence of GXF tables is also verified, a step that will be revisited in further detail. The translator scans all valid files within the provided superset directory and initially filters them by removing those that appear to be readme (e.g. dataset description) files from the translation queue. Additional filtration informs the translator to skip files that do not constitute completed RNA-seq analysis results, with the following formats being removed: BED, BedGraph, Wiggle, BigWig (e.g. binary Wiggle), SAM, BAM, FASTA, XML, PDF, and JSON. The removal of these file formats from GEORGET's translation queue is based upon the definition of finalized RNA-seq results asserted by the MINSEQE standards, which—in major section 3—require that transcriptomics results be clearly presented in matrix format [29]. While BED and Wiggle formats and their derivatives are technical matrices, they do not contain expression information and are commonly used for display of read pile-up alignments, ideally visualized by genome browsers. XML and JSON formats, while being ubiquitous standards elsewhere in computer science and informatics, are tree-based data structure formats that do not satisfy the MINSEQE requirement.

Files that pass the format filter are then tested by GEORGET for their filesystem size; if this exceeds a specified threshold, or 500 megabytes by default, the file is skipped during translation. This filter exists to prevent attempted translation of RNA-seq results that are read-specific, which are too premature within an RNA-seq analysis pipeline to be considered amenable [or desirable] for database translation. GEORGET then detects the column separator of filter-passing files and surveys these files for their column-count uniformity; that is, the number of columns identified per line throughout the files' contents. If a file is found to be entirely homogeneous with respect to column-count, it is placed through one final filter: if the homogeneous column-count equals one, the file is discarded; if the file's column-

count equals nine and the file is identified as a GFF or GTF file [30], it is translated as a GXF file; otherwise, the file is translated and database-loaded as a uniform RNA-seq results file.

Handling of results files possessing a heterogeneous column-count is more involved. First, the data structure containing column-counts and their associated frequencies is appended with their percentage compositions, and the top-most result is passed through a specified ambiguity threshold. If this percentage falls below the threshold, or 95% by default, the file is discarded from the translation queue. Next, if the file's majority column-count equals nine and the file passes the GFF specification, it is translated as a series of GXF entries. If a heterogeneous file arrives at this point without being identified as a GXF, it is scanned for the first tuple that likely identifies the attributes (e.g. columns) for the remainder of the file. A common issue we have experienced with ambiguous results files is that their attribute definition line is often one column short of the majority column-count for the remainder of the file, and this missing column is likely the feature identification column (e.g. gene, transcript, or feature ID). Wherever possible, GEORGET attempts to re-insert this column identifier when needed; if this is found to be impossible, the translator discards the ambiguous file, as it is extremely difficult to accurately reconstruct column identifiers that match what researchers may have originally intended.

Following translation and database loading, GEORGET optionally indexes the populated RNA-seq results tables using the SQLite temporary store pragma either set by the user or through use of system memory by default. Finally, GEORGET reports its raw and adjusted translation metrics; the adjusted translation rate is computed by excluding previously mentioned file formats from the total count of results files. The GEORGET workflow has been implemented in Bourne shell (e.g. Bash), GNU Parallel [27], and Modern Perl, with a requirement for the DBI and DBD::SQLite Perl modules to be installed in order to run. Although installation of GNU Parallel is optional, its use is highly recommended to significantly expedite pre-translation preparatory steps.

Results and Discussion

To assess the efficacy of GEORGET, we designed two randomized RNA-seq superset trials in addition to the translator's approximately 130-study zebrafish RNA-seq training superset; these random supersets were pulled from the human and mouse stores of RNA-seq studies, each of which currently house thousands of RNA-seq studies. The effective translation rate of GEORGET was found to be 585 / 640 files (91.41%) for the zebrafish training superset, 403 / 411 files (98.05%) for the randomized mouse superset, and 924 / 992 files (93.15%) for the randomized human superset. While observation of these initial results has been gratifying, the translation model has significant room for continued growth. As further RNA-seq studies are translated, GEORGET will grow to encompass increasing numbers of potential file formats. While it is certain that a theoretically infinite number of RNA-seq results formats may exist, the translation model behind GEORGET—in concert with the existing MINSEQE standard—allows for translation of a pseudo-infinite subset number of results formats that are, at minimum, matrix-compliant. This subset facilitates and encourages establishment of a domain of MINSEQE-permitted standards, rather than a single narrow standard. Future work with GEORGET will address incorporation of variant-calling data and may extend to translation of genomic histogram results (e.g. BED- and Wiggle-derived formats). The primary focus of this work in the future will be the establishment of a Web portal for user exploration of translated RNA-seq results.

In considering the scope of GEORGET, one might question how this results translation workflow is not unlike classical "screen scraping", a well-described frustration for the bioinformatics community throughout the past two decades [31]. In screen scraping, the informatician generally constructs a form of Web crawler that loads database pages served through a Web resource that extracts the HTML source of the page; this source mark-up is then parsed for desired information for use in downstream applications. This process is inherently fragile, as front-end developers may modify Web resources' user interfaces; this subsequently modifies the page HTML, which requires the informatician to re-factor

their screen scraper in order to pull and extract the desired information once again. Additionally, database administrators may optimize back-end data structures, which may also result in cascading changes to front-end HTML indirectly, requiring the informatician to backtrack yet again. In contrast, RNA-seq results deposited to GEO often belong to studies whose results are thereafter published; the structures of these datasets are significantly more static than those that are frequently modified from Web resources lacking a stable application program interface (API). Of crucial importance is the fundamental delineation in structure between screen scraping targets and RNA-seq results stored within GEO: Web-based structures are tree-like in nature, yet RNA-seq results, which ideally conform to the MINSEQE specification, are presented as matrices.

Summary

The workflow described here, GEORGET, represents an effort to transparently and reproducibly extract, translate, and load RNA-seq results stored within the Gene Expression Omnibus (GEO) into a relational database store. This store is currently implemented using SQLite but may be changed in the future to support larger supersets of RNA-seq data and with greater translational concurrency. To our current knowledge, GEORGET appears to be the first workflow of its kind in attempting to translate deposited RNA-seq analysis results. Existing gold-standard alternatives specialize in secondary re-analysis, which—while essential for future statistical accuracy and interoperability between datasets—may be further augmented with the context of original analyses that are published into existing scientific literature. Ideally, we envision a future where output from GEORGET may be integrated with existing gene expression atlases.

References

1) Schuster S C. Next-generation sequencing transforms today's biology. Nature Methods. 2008; 5: 16-18.

2) Mardis E R. The impact of next-generation sequencing technology on genetics. Trends in Genetics. 2008; 24 (3): 133-141.

3) Metzker M L. Sequencing technologies — the next generation. Nature Reviews Genetics. 2010; 11: 31-46.

4) Morozova O and Marra M A. Applications of next-generation sequencing technologies in functional genomics. Genomics. 2008; 92 (5): 255-264.

5) Reis-Filho J S. Next-generation sequencing. Breast Cancer Research. 2009; 11 (S3): S12.

6) Korbel J O, Urban A E, Affourtit J P et al. Paired-End Mapping Reveals Extensive Structural Variation in the Human Genome. Science. 2007; 318 (5849): 420-426.

7) Quail M A, Smith M, Coupland P et al. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics. 2012; 13: 341.

8) Marguerat S and Bähler J. RNA-seq: from technology to biology. Cellular and Molecular Life Sciences. 2010; 67 (4): 569-579.

9) Wang Z, Gerstein M, and Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics. 2009; 10: 57-63.

10) Oshlack A, Robinson M D, and Young M D. From RNA-seq reads to differential expression results. Genome Biology. 2010; 11: 220.

11) Trapnell C, Pachter L, and Salzberg S L. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics. 2009; 25 (9): 1105-1111.

12) Wu T D and Nacu S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. Bioinformatics. 2010; 26 (7): 873-881.

13) Dobin A, Davis C A, Schlesinger F et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics. 2013; 29 (1): 15-21.

14) Trapnell C, Roberts A, Goff L et al. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nature Protocols. 2012; 7: 562-578.

15) Quinlan A R and Hall I M. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010; 26 (6): 841-842.

16) Anders S, Pyl P T, and Huber W. HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics. 2015; 31 (2): 166-169.

17) Li B and Dewey C N. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics. 2011; 12: 323.

18) Robinson M D, McCarthy D J, and Smyth G K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics. 2010; 26 (1): 139-140.

19) Anders S and Huber W. Differential expression of RNA-Seq data at the gene level – the DESeq package. European Molecular Biology Laboratory (EMBL). 2012.

20) Bray N L, Pimentel H, Melsted P, and Pachter L. Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology. 2016; 34: 525-527.

21) Patro R, Mount S M, and Kingsford C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nature Biotechnology. 2014; 32: 462-464.

22) Ben-Ari Fuchs S, Lieder I, Stelzer G et al. GeneAnalytics: An Integrative Gene Set Analysis Tool for Next Generation Sequencing, RNAseq and Microarray Data. OMICS. 2016; 20 (3): 139-151.

23) Pimentel H, Sturmfels P, Bray N et al. The Lair: a resource for exploratory analysis of published RNA-Seq data. BMC Bioinformatics. 2016; 17: 490.

24) Pimentel H, Bray N L, Puente S et al. Differential analysis of RNA-seq incorporating quantification uncertainty. Nature Methods. 2017; 14: 687-690.

25) Papatheodorou I, Fonseca N A, Keays M et al. Expression Atlas: gene and protein expression across multiple studies and organisms. Nucleic Acids Research. 2017; gkx1158.

26) Barrett T, Wilhite S E, Ledoux P et al. NCBI GEO: archive for functional genomics data sets—update. Nucleic Acids Research. 2013; 41: D991-D995.

27) Tange O. GNU Parallel—the command-line power tool. The USENIX Magazine. 2011.

28) Zhang Y, Liu T, Meyer C A et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biology. 2008; 9: R137.

29) MINSEQE: Minimum Information about a High-throughput Sequencing Experiment. The Functional Genomics Data Society (FGED). 2012. http://fged.org/site_media/pdf/MINSEQE_1.0.pdf. Accessed July 2017.

30) Stein L. Generic Feature Format Version 3 (GFF3). The Sequence Ontology. 2013. https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md. Accessed July 2017.

31) Stein L. Creating a bioinformatics nation. Nature. 2002; 417: 119-120.

Chapter 5: Conclusion

Recapitulation

The purpose behind this work has been to devise a workflow in which RNA-seq atlases can be reproducibly and transparently constructed from metadata and supplemental expression analysis results stored within the Gene Expression Omnibus. Expression atlases are of great use to researchers devising future research questions; by presenting gene expression results from studies concerning species and experimental conditions of interest, atlas users can explore a kaleidoscope of scientific avenues for future work and may observe expression patterns integral to research question formation. While the European RNA-seq Expression Atlas serves as an established gold-standard for modern biomedical literature, it forcibly employs secondary re-analysis of studies without regard for original results submitted to GEO, resulting in potential discordance between sufficiently well-analyzed RNA-seq results published within the literature and secondary results presented on the Web by the Expression Atlas. GEORAC aims to bridge this gap between GEO and the Expression Atlas, such that submitted RNA-seq results may ideally be presented in addition to secondary re-analysis results.

Chapter 2 from this work concerns the development and implementation of GEOMP, a metadata parser and relational database constructor for GEO. Canonically, GEOMP has been preceded by GEOquery, an R-based package for pulling and parsing GEO metadata for use in downstream microarray applications within the R Bioconductor framework of biostatistics packages. GEOMP advances past the preexisting features of GEOquery in numerous ways: while GEOquery requires users to specify individual GEO accession IDs to pull and parse study and sample metadata, GEOMP permits users to provide a query taken directly from the NCBI GEO Web interface, allowing users to jump directly from query to normalized metadata stores; this process conserves user effort on the order of hours to days. Further, GEOMP accepts user-defined lists of metadata attributes to normalize and

output in various formats; users may even opt to allow the tool to automatically select a list of chosen metadata attributes, ordered by frequency of occurrence, with up to 100% of all occurring attributes available for normalization and subsequent use. Finally, GEOMP allows users to create a SQLite relational database on demand and may additionally output tables containing uniquely occurring bioinformatics methods by study.

Chapter 3 aims to make significant use of the bioinformatics methods tables produced by GEOMP; reproducibility in biomedical research has long been an increasing concern harbored by scientific funding agencies, with the executive director of the NIH authoring a piece within the past half-decade calling for improved reproducibility measures and accountability within biomedical research. To address this critical concern, RNA-seq analysis methods documented within GEO were assessed against a five-point rubric for reproducibility among all zebrafish RNA-seq experiments submitted to GEO up to early 2017. Because the mouse and human research communities are orders of magnitude larger than the rat and zebrafish communities, randomized supersets numbering approximately 100 studies from each community were chosen for metadata normalization and subsequent bioinformatics methods assessment. Results from this pilot study were found to be both surprising and dire: human RNA-seq research appeared to lag behind that of the zebrafish and mouse communities in terms of its capacity to be replicated. In a further concerning finding, among nearly 300 studies assessed by this work, none were found to report hardware runtime requirements for computationally significant RNA-seq workflows. This chapter also presents GEOMP2, an expansion upon its preceding work, in which atomization of GEO sample characteristic tags has been enhanced, such that individual tags [and their corresponding values] may be directly queried from user-generated SQLite databases.

Chapter 4 presents a functional prototype implementation of GEORGET, an RNA-seq supplemental results translator for the Gene Expression Omnibus; this work details a multi-step pipeline in which supplemental expression analysis results are pulled from the GEO FTP store and are

subsequently quality-controlled in preparation for dynamic translation and loading into the relational database(s) created via GEOMP2. The translator primarily functions by scanning a given RNA-seq results file and, where possible, establishes a constant column-count for the apparent matrix; this assumption is based upon the latest release of the MINSEQE standard. If the attribute definition line of the results file is found to be in good agreement with the remainder of the file's tabular contents, then the file is translated and loaded into the target GEORAC database. GEORGET additionally handles translation and database loading of GFF and GTF (e.g. gene transfer format) formats, with these formats being given their own type of translated results table within the resulting GEORAC database. To test the translator, GEORGET was assessed against the zebrafish, human, and mouse RNA-seq supersets from the preceding chapter; the effective translation rate of RNA-seq supplemental results by GEORGET was found to be 91%, 93%, and 98%, respectively.

Future Work

Despite the aforementioned successes with prototypical development and implementation of the GEORAC workflow, significant room exists to continue enhancing both the flexibility and scalability of this RNA-seq atlas construction framework. With respect to GEOMP2, there remains much room for improvement: despite early signs of maturity (e.g. capacity to normalize 100% of encountered series and sample metadata attributes), the tool continues to lack an effective mechanism for updating large metadata repositories. Currently, the most viable updating approach is to pull an entire search query's associated metadata from GEO, parse the contents, and re-instantiate a new or updated SQLite database from the pulled contents. While this method remains functional for smaller research communities, such as zebrafish or rat, this approach quickly degrades in terms of performance for communities that are orders of magnitude larger in size, such as mouse or human. Future work with GEOMP2 will primarily aim to add the capacity to pull metadata by submission or update dates from

GEO, in addition to the ability to integrate newer metadata stores with existing repositories. Additional areas for significant growth include database handling of parsed GEO [sequencing] platforms and introductory support for super-series metadata; the latter of these refers to metadata for collections of studies.

The bioinformatics methods reproducibility pilot study may be significantly improved upon and further expanded. Initial results presented here, while edifying, were generated using a single methods assessor; recruitment of additional judges will add weight to the significance of the study's findings, and it can be expected that further statistical metrics, such as the computing of study kappa values, will quantify the measure of agreement among judges. Given the limited scope presented here, this reproducibility study can be expanded to an even larger scope in the future, such that many hundreds of studies' methods may be assessed. This would further strengthen the finding reported here: human research, which is integral to the design of clinical trials and the application of precision and personalized medicine, requires much greater discipline in terms of documenting biomedical analysis methods. A further avenue of inquiry relates to the comparison between bioinformatics methods submitted to GEO versus their analogous expositions within the literature; this work would aim to determine whether they differ, and if so, how significantly they differ.

Functionality of GEORGET's prototype implementation, while impressive, leaves enormous room for future revisions and expansion. Currently, the translator operates in an unsupervised fashion, generating dense matrices of rehabilitated and translated supplemental RNA-seq results within expression tables of the resulting GEORAC database. This approach works sufficiently well for smaller RNA-seq research communities but degrades rapidly in terms of performance as the input data store increases exponentially in size. Consider the three supersets previously described by the reproducibility study: these same three supersets were employed for the stress-testing of GEORGET's functionality, and both the human and mouse experiments—while successful in terms of results translation—produced

SQL databases on the order of hundreds of gigabytes in filesystem footprint size. This may seem appropriate, but these were only small slices of the true RNA-seq supersets for these research communities, with studies numbering in the many thousands. Thus, a species-wide RNA-seq atlas for the mouse or human communities would likely result in a database on the order of many terabytes, and at such scale of “big data”, the performance of structured query language (SQL) databases begins to degrade significantly, requiring a potential shift in database archetype to NoSQL databases.

An alternative path to consider in escaping the translation quandary: a radical re-design of the GEORGET translation workflow, such that the translation process would become largely curation-based. This would result in effective translation rates approaching 100% and would also limit the produced GEORAC expression database to one that is minimally sparse in terms of matrix format, resolving the database architecture challenge. In such a scenario, it may be possible to imagine designing a machine learning-like feedback process by which the work of bioinformatics results curators may ideally be assisted or expedited through machine-learned patterns observed from RNA-seq datasets.

Closing Remarks

The problem domain addressed in part by this work is, without question, a gargantuan one; parallel efforts made by the Expression Atlas project over the past decade clearly demonstrate that well over a century’s worth of manpower can be easily invested into a research aim of this magnitude. The GEORAC project has been, and will continue to be, a divergent effort to resolve the challenges associated with accessing and improving upon the structure- and standard-less metadata and supplemental expression analysis results archived within the Gene Expression Omnibus. GEORAC aims to bridge the gap between literature-reported results from original scientific work and secondary re-analyses made public to the Web by the Expression Atlas; in doing so, the future of this project lies with

making publicly available a curated subset of complete, or partially complete, RNA-seq results for numerous species of scientific interest.

Appendix A: Licensing

License Description

This work is distributed under version 4.0 of the international Creative Commons (CC) BY-NC-ND license. Given these terms, any future work incorporating content provided here must appropriately attribute credit to the original licensor of this work. The material provided here may not be used for commercial or for-profit applications or purposes. Future work that expands upon or transforms the work described here, or that is partially or fully derived from the present work in any manner, is expressly prohibited; derivatives of this work may not be distributed in any public manner. Users of the GEORAC framework interested in bug fixes and feature requests are strongly encouraged to address these needs through direct contact with the author of this work. A formal copy of the license follows.

Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.
b. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.
c. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.
d. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.
e. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.
f. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.
g. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

h. **NonCommercial** means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.
i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.
j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.
k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope

a. **License grant**
    1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
        A. reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and
        B. produce and reproduce, but not Share, Adapted Material for NonCommercial purposes only.
    2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
    3. Term. The term of this Public License is specified in Section 6(a).
    4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.
    5. Downstream recipients.
        A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
        B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.
    6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is,

connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. **Other rights**
    1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
    2. Patent and trademark rights are not licensed under this Public License.
    3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes.

Section 3 – License Conditions

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. **Attribution**
    1. If You Share the Licensed Material, You must:
        A. retain the following if it is supplied by the Licensor with the Licensed Material:
            i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
            ii. a copyright notice;
            iii. a notice that refers to this Public License;
            iv. a notice that refers to the disclaimer of warranties;
            v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
        B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
        C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

        For the avoidance of doubt, You do not have permission under this Public License to Share Adapted Material.
    2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
    3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

Section 4 – Sui Generis Database Rights

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only and provided You do not Share Adapted Material;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and
c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.
b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:
    1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
    2. upon express reinstatement by the Licensor.

    For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.
c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.
d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.
b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.
b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.
c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.
d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Appendix B: GEORAC Source Code

geomp_pull

```
#!/usr/bin/env bash
# Aurash

################# DOCUMENTATION #################
# This Bash script is designed to
# obtain uIDs from NCBI for all
# RNA-seq studies related to the
# specified species, convert those
# uIDs into GEO accession IDs, and
# then fetch the associated GEO SOFT
# format files.
#
# SOFT files specifying SuperSeries
# entries in GEO are then deleted,
# while valid SOFTs are truncated
# by removing platform-to-sample
# associations.
#
# Usage example:
# ./geomp_pull "Danio rerio" | tee pullGEO.log
#################################################

if [[ $# -lt 2 || ${1} == "-h" || ${1} == "--help" ]]
then
	printf "\nUsage: [./]geomp_pull \"GEO Web query\" outputDirectory
maxCount\n\n"
	exit
fi

query=${1}
outputDir=${2}
maxCount=${3}

if [[ ${query} =~ ^RNA-seq= ]]
then
	query=$(printf "${query}" | sed 's/^RNA-seq=//; s/ /+/g;')
	query="Expression+profiling+by+high+throughput+Sequencing[DataSet+Type]+AND+${
query}[Organism]"
else
	query=$(printf "${query}" | sed 's/ /+/g')
fi

if [[ ${maxCount} == "" ]]
then
	maxCount=300
fi

query="${query}&retmax=${maxCount}"
```

```
# Sanity check
# printf "\nquery: ${query}\noutputDir: ${outputDir}\n\n"
# exit

# Use NCBI eSearch to obtain the NCBI
# uIDs for the first 1,000 studies
# found within GEO
printf "\nObtaining NCBI uIDs ... "
wget -q -O -
"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=gds&term=${query}" |
grep "^\s*<Id>[0-9]*</Id>" | sed 's/^\s*<Id>\|<\/Id>.*$//g' > eSearch.log
printf 'done!\n'

if [[ $(wc -l < eSearch.log) -eq 0 ]]
then
	printf 'No uIDs found!\n'
	exit
fi

# Check for the existence of the SOFT
# sub-directory; if it doesn't exist,
# create it
if [[ ! -d ${outputDir} ]]
then
	mkdir ${outputDir}
	printf "SOFT sub-directory generated: ${outputDir}/\n"
fi

# For each NCBI uID pulled...
while read id
do
	# find the corresponding GEO
	# accession identifier...
	acc=$(wget -q -O -
"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=gds&id=${id}" | grep -o
"Accession: GSE[0-9]*" | sed 's/^Accession: //')

	# and pull its associated
	# SOFT file
	printf "Pulling ${id} as ${acc}.soft ... "
	wget -q -O ${outputDir}/${acc}.soft
"https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=${acc}&targ=all&view=brief&form=t
ext"
	printf 'done!\n'
done < eSearch.log
```

```
# Move into the SOFT sub-directory
pushd ${outputDir}

# For every SOFT file in the local
# directory...
for file in *.soft
do
	# Determine if the given SOFT
	# file specifies a SuperSeries
	count=$(grep "^\!Series_summary = This SuperSeries" ${file} | wc -c)

	if [[ ${count} -gt 0 ]]
	then
		# Remove SuperSeries SOFT files
		rm ${file}
		printf "SuperSeries removed: ${file}\n"
	else
		# Modify remaining (e.g. valid)
		# SOFT files to not specify
		# samples associated with
		# experimental platforms and
		# report on the reduction in
		# size for each SOFT file
		startLines=$(wc -l < ${file})
		sed '/^!Platform_sample_id/d' ${file} > tmp
		mv tmp ${file}
		endLines=$(wc -l < ${file})
		reduction=$(printf "%.2f" $(printf "(${startLines} - ${endLines}) /
${startLines} * 100\n" | bc -l))
		printf "${file}: ${reduction}%% reduction from ${startLines} to
${endLines} lines\n"
	fi
done

# Move back to the parent directory
popd
printf 'All done!\n\n'
```

geomp2

```
#!/usr/bin/env perl
# Aurash

# Pragmas
use strict;
use warnings;
use autodie;

# Variable declarations
my ( %gse, %gseAttr, @printGSEattr, %gsm, %gsmAttr, @printGSMattr, %gpl, %gplAttr,
@printGPLattr, %char, %charAttr, @printCHARattr ) = ();
my ( $curSeries, $curPlatform, $curSample ) = ();
my ( $autoConfig, $configFile, $softDir, $targetOrganism, $defaultEmptyString,
$outputFile, $methodsOutfile, $dbFile, $charConfig, $tagsFile ) = ( 'N95', ( undef )
x 3, 'NA', ( undef ) x 5 );

# Configuration dispatch tables
my %doubleFlagParams =
(
	auto => { var => \$autoConfig, default => 'N95' },
	config => { var => \$configFile, default => undef },
	dir => { var => \$softDir, default => undef },
	species => { var => \$targetOrganism, default => undef },
	null => { var => \$defaultEmptyString, default => 'NA' },
	out => { var => \$outputFile, default => '' },
	methods => { var => \$methodsOutfile, default => '' },
	db => { var => \$dbFile, default => '' },
	tags => { var => \$charConfig, default => 'N95' },
	tagsFile => { var => \$tagsFile, default => '' }
);
my %singleFlagParams =
(
	a => { var => \$autoConfig, default => 'N95' },
	c => { var => \$configFile, default => undef },
	d => { var => \$softDir, default => undef },
	s => { var => \$targetOrganism, default => undef },
	n => { var => \$defaultEmptyString, default => 'NA' },
	o => { var => \$outputFile, default => '' },
	m => { var => \$methodsOutfile, default => '' },
	b => { var => \$dbFile, default => '' },
	t => { var => \$charConfig, default => 'N95' },
	T => { var => \$tagsFile, default => '' }
);
```

```
# Functions
sub valid
{
	my $var = $_[ 0 ];

	if( defined $var && $var ne '' ) { return 1; }
	else { return 0; }
}

sub autosetVar
{
	my ( $refVar, $compVar, $append ) = @_;

	if( defined $$refVar && $$refVar eq '' )
	{
		( $$refVar = $compVar ) =~ s/_SOFT$//;
		$$refVar .= $append;
	}
}

sub parseArgs
{
	for my $i ( 0 .. $#ARGV )
	{
		if( $ARGV[ $i ] =~ m/^--/ ) { parseDoubleFlagArg( $i ); }
		elsif( $ARGV[ $i ] =~ m/^-/ ) { parseSingleFlagArg( $i ); }
	}

	die "No directory of SOFT files specified" unless( valid( $softDir ) );

	$softDir =~ s/\/+$//;
	autosetVar( \$outputFile, $softDir, '_geomp.tsv' );
	autosetVar( \$methodsOutfile, $softDir, '_methods.tsv' );
	autosetVar( \$tagsFile, $softDir, '_tags.tsv' );
	autosetVar( \$dbFile, $softDir, '.db' );

	directoryReader();
	countSeriesOrganisms();
	populateCharTags();

	if( valid( $configFile ) ) { configReader(); }
	else { autogenConfig(); }

	normalizeGEO();

	printGEO() if( valid( $outputFile ) );

	# Requests from tp5:
	# Print [unique] data processing methods
	printDataProcMethods() if( valid( $methodsOutfile ) );
	# Print atomized sample characteristic tags
	printCharTags() if( valid( $tagsFile ) );

	generateDB() if( valid( $dbFile ) );
}
```

```
sub parseDoubleFlagArg
{
	my ( $i, $arg, $value ) = ( $_[ 0 ], $ARGV[ $_[ 0 ] ], undef );
	$arg = substr $arg, 2;
	( $arg, $value ) = split '=', $arg;

	if( $arg eq 'help' ) { printHelp(); exit; }

	if( defined $doubleFlagParams{ $arg } )
	{
		if( valid( $value ) )
		{
			${ $doubleFlagParams{ $arg }{ var } } = $value;
		}
		else
		{
			${ $doubleFlagParams{ $arg }{ var } } = $doubleFlagParams{ $arg
}{ default };
		}
	}
	else { die "Unrecognized argument: --${arg}"; }
}

sub parseSingleFlagArg
{
	my ( $i, $arg, $value ) = ( $_[ 0 ], $ARGV[ $_[ 0 ] ], undef );
	$arg = substr $arg, 1;
	$value = $ARGV[ $i + 1 ] if( defined $ARGV[ $i + 1 ] && $ARGV[ $i + 1 ] !~
m/^-/ );

	if( $arg eq 'h' ) { printHelp(); exit; }

	if( defined $singleFlagParams{ $arg } )
	{
		if( valid( $value ) )
		{
			${ $singleFlagParams{ $arg }{ var } } = $value;
		}
		else
		{
			${ $singleFlagParams{ $arg }{ var } } = $singleFlagParams{ $arg
}{ default };
		}
	}
	else { die "Unrecognized argument: -${arg}"; }
}
```

```
sub printHelp
{
	print << 'EOF';

GEOMP2: an Automated Metadata Parser and Relational Database Constructor for
	the Gene Expression Omnibus

	by Aurash Mohaimani

DESCRIPTION:
	GEOMP2 is a program designed to parse metadata anchors and attributes
	from the Simple Omnibus Format in Text (SOFT) format exported from
	the Gene Expression Omnibus (GEO) via the Entrez E-Utils and GEO URL
	construction APIs. GEOMP2 additionally parses through the unique set
	of bioinformatics methods employed by each study and may also be used
	to build an atomized relational database, given a user-defined
	configuration file of desired GEO metadata attributes. If no config
	file is available, GEOMP2 can automatically construct one. GEOMP2 is
	capable of atomizing GEO sample (e.g. GSM) characteristic tags for
	optional printing to a file and/or loading into a database.

USAGE:
	[./]geomp2 -d softDirectory [-a[ autoConfig] OR -c configFile]
	[-s "species"] [-t[ autoTagsConfig] [-n defaultEmptyString]
	[-o[ outFile]] [-m[ methodsFile]] [-T[ tagsFile]] [-b[ dbFile]]

	OR

	[./]geomp2 --dir=softDirectory [--auto[=autoConfig] OR
	--config=configFile] [--species="species"] [--tags[=autoTagsConfig]]
	[--null=defaultEmptyString] [--out[=outFile]] [--methods[=methodsFile]]
	[--tagsFile[=tagsFile]] [--db[=dbFile]]

OPTIONS:
	-d softDirectory, --dir=softDirectory
		REQUIRED: specifies the path to the directory containing the
		target *.SOFT files.

	-a[ autoConfig], --auto[=autoConfig]
		Instructs GEOMP2 to automatically generate and use a
		configuration file. By default, GEOMP2 will aim to include
		metadata attributes that constitute the N95 of the chosen
		dataset. Percentage values may alternatively be used.
		NOTE: choose either this option or the manual config file
		option below! (default: 'N95')

	-c configFile, --config=configFile
		Specifies the path to the configuration file containing the
		GEO series, sample, and characteristic tag attributes that will
		be parsed and loaded as columns of output data. NOTE: specify
		either this option or the autoConfig option above!
```

```
    -s "species", --species="species"
           Defines the full name of the species to parse; this is the same
           string used in GEO search queries (examples: "Danio rerio",
           "Homo sapiens", etc). Providing this will allow GEOMP2 to
           generate the "targetOrganismCount" series attribute.

    -t[ autoTagsConfig], --tags[=autoTagsConfig]
           Instructs GEOMP2 to automatically construct a list containing
           the most frequently occurring sample characteristics tags
           according to the given N- or percentage-based threshold.
           (default: 'N95')

    -n defaultEmptyString, --null=defaultEmptyString
           Defines the normalization string to use for an undefined
           metadata attribute. (default: 'NA')

    -o[ outFile], --out[=outFile]
           Enables write-out of parsed metadata to a tab-delimited file.
           If no file name is provided, the output file will be generated
           using the given directory name. (default: undefined)

    -m[ methodsFile], --methods[=methodsFile]
           Enables write-out of tab-delimited bioinformatics methods data
           to the given file name. If no file name is provided, the file
           will be generated using the given directory.
           (default: undefined)

    -T[ tagsFile], --tagsFile[=tagsFile]
           Enables printing of atomized sample characteristic tags to the
           specified file. If no file name is provided, then the file name
           will be generated using the given directory.
           (default: undefined)

    -b[ dbFile], --db[=dbFile]
           Enables generation of an atomized SQLite database containing
           tables corresponding to GEO series and sample entries, with
           additional tables generated for metadata attributes possessing
           cardinality of the form 1:N. Attributes are specified by
           the configuration file or can be automatically chosen via the
           -a and -t flags. NOTE: installation of the DBD::SQLite module
           is REQUIRED to enable this option!

    -h, --help
           Prints this help page.

EOF
}
```

```
sub directoryReader
{
	opendir( my $dh, $softDir );

	while( my $file = readdir( $dh ) )
	{
		next unless( $file =~ m/\.soft$/ );

		fileReader( $file );
	}

	closedir( $dh );
}

sub fileReader
{
	my ( $file, $firstChar ) = ( $_[ 0 ], undef );

	open( my $fh, '<', "$softDir/$file" );

	while( my $line = <$fh> )
	{
		chomp $line;
		$line =~ s/\s+$//;
		$firstChar = substr $line, 0, 1;

		if( $firstChar eq '^' ) { anchorReader( $line ); }
		elsif( $firstChar eq '!' ) { attributeReader( $line ); }
	}

	close $fh;
}

sub anchorReader
{
	my ( $type, $id ) = split ' = ', $_[ 0 ];
	$type = substr $type, 1;

	if( $type eq 'SERIES' )
	{
		$curSeries = $id;
	}
	elsif( $type eq 'PLATFORM' )
	{
		if( defined $gpl{ $id } ) { $curPlatform = undef; }
		else { $curPlatform = $id; }
	}
	elsif( $type eq 'SAMPLE' )
	{
		$curSample = $id;
	}
}
```

```
sub attributeReader
{
	my $line = substr $_[ 0 ], 1;
	my ( $attrName, $attrValue ) = split ' = ', $line;
	my $ref;

	return unless( defined $attrValue );

	if( $line =~ m/^Series/ )
	{
		$attrName = substr $attrName, 7;
		$gseAttr{ $attrName }{ count } += 1;
		$ref = \$gse{ $curSeries }{ $attrName };
	}
	elsif( $line =~ m/^Platform/ )
	{
		return unless( defined $curPlatform );

		$attrName = substr $attrName, 9;
		$gplAttr{ $attrName }{ count } += 1;
		$ref = \$gpl{ $curPlatform }{ $attrName };
	}
	elsif( $line =~ m/^Sample/ )
	{
		$attrName = substr $attrName, 7;
		$attrName = 'supplementary_file' if( $attrName =~
m/^supplementary_file_[0-9]+/ );
		$gsmAttr{ $attrName }{ count } += 1;
		$ref = \$gsm{ $curSample }{ $attrName };
	}

	push @{ $$ref }, $attrValue;
}

sub getAttrText
{
	my $field = $_[ 0 ];

	if( ref $field eq 'ARRAY' ) { return join( '||', @{ $field } ); }
	else { return $field; }
}
```

```
sub countSeriesOrganisms
{
	foreach my $series ( keys %gse )
	{
		$gseAttr{ totalOrganismCount }{ count } += 1;
		$gseAttr{ targetOrganismCount }{ count } += 1 if( defined
$targetOrganism );

		foreach my $sample ( @{ $gse{ $series }{ sample_id } } )
		{
			$gse{ $series }{ totalOrganismCount } += 1;
			$gse{ $series }{ targetOrganismCount } += 1 if( defined
$targetOrganism && getAttrText( $gsm{ $sample }{ organism_ch1 } ) eq $targetOrganism
);
		}
	}
}

sub populateCharTags
{
	my ( $ref, $tag, $value ) = ( ( undef ) x 3 );

	foreach my $sample ( keys %gsm )
	{
		$ref = \$gsm{ $sample }{ characteristics_ch1 };

		next unless( defined $$ref && ref $$ref eq 'ARRAY' );

		foreach my $line ( @{ $$ref } )
		{
			( $tag, $value ) = split /:\s+/, $line;

			next unless( defined $value );

			$char{ $sample }{ $tag } = $value;
			$charAttr{ $tag } += 1;
		}
	}
}
```

```
sub configReader
{
	my $ref;

	open( my $fh, '<', $configFile );

	while( my $line = <$fh> )
	{
		chomp $line;

		next if( $line =~ m/^\s*$|^\s*#/ );

		if( $line =~ m/^\s*\[/ )
		{
			if( $line =~ m/^\s*\[\s*GSE\s*\]/ ) { $ref = \@printGSEattr; }
			elsif( $line =~ m/^\s*\[\s*GSM\s*\]/ ) { $ref = \@printGSMattr; }
			elsif( $line =~ m/^\s*\[\s*CHAR\s*\]/ ) { $ref = \@printCHARattr;
}
		}
		else { push @$ref, $line; }
	}

	close $fh;
}

sub autogenConfig
{
	my $cutoff = setAutoConfig( $autoConfig ) / 100;

	autosetAttrs( $cutoff, \%gseAttr, \@printGSEattr );
	autosetAttrs( $cutoff, \%gsmAttr, \@printGSMattr );
	autosetCharAttrs() if( valid( $charConfig ) );

	writeAutoConfig( 'auto.conf' );
}

sub setAutoConfig
{
	my $param = $_[ 0 ];

	if( $param =~ m/^[Nn][0-9]+$/ ) { $param =~ s/^[Nn]//; return $param; }
	elsif( $param =~ m/^[0-9]*(\.[0-9]*)?%$/ ) { $param =~ s/%$//; return $param;
}
	else { die "Invalid \$autoConfig parameter: $param"; }
}
```

```
sub autosetAttrs
{
	my ( $cutoff, $attrs, $printAttrs ) = @_;
	my ( $total, $count, $attr ) = ( ( 0 ) x 2, undef );
	map { $total += $$attrs{ $_ }{ count }; } keys %$attrs;
	$cutoff = int( $cutoff * $total - 1 );

	foreach my $attr ( sort { $$attrs{ $b }{ count } <=> $$attrs{ $a }{ count } ||
$a cmp $b } keys %$attrs )
	{
		push @$printAttrs, $attr;
		$count += $$attrs{ $attr }{ count };

		last if( $count > $cutoff );
	}

	for my $i ( 0 .. $#$printAttrs )
	{
		$attr = $$printAttrs[ $i ];

		if( $attr eq 'geo_accession' )
		{
			splice @$printAttrs, $i, 1;
			unshift @$printAttrs, $attr;
		}
	}
}

sub autosetCharAttrs
{
	my $cutoff = setAutoConfig( $charConfig ) / 100;
	my ( $total, $count, $attr ) = ( ( 0 ) x 2, undef );
	map { $total += $charAttr{ $_ }; } keys %charAttr;
	$cutoff = int( $cutoff * $total - 1 );

	foreach my $attr ( sort { $charAttr{ $b } <=> $charAttr{ $a } || $a cmp $b }
keys %charAttr )
	{
		push @printCHARattr, $attr;
		$count += $charAttr{ $attr };

		last if( $count > $cutoff );
	}
}
```

```
sub writeAutoConfig
{
	my $file = $_[ 0 ];

	open( my $fh, '>', "$softDir/$file" );

	print $fh "[ GSE ]\n";
	map { print $fh "$_\n"; } @printGSEattr;
	print $fh "\n[ GSM ]\n";
	map { print $fh "$_\n"; } @printGSMattr;
	print $fh "\n[ CHAR ]\n";
	map { print $fh "$_\n"; } @printCHARattr;

	close $fh;
}

sub normalizeGEO
{
	normalizeData( \%gse, \@printGSEattr );
	normalizeData( \%gsm, \@printGSMattr );
	normalizeData( \%char, \@printCHARattr ) if( scalar @printCHARattr > 0 );

	setCardinality( \%gse, \%gseAttr, \@printGSEattr );
	setCardinality( \%gsm, \%gsmAttr, \@printGSMattr );
}

sub normalizeData
{
	my ( $data, $attrs ) = @_;

	foreach my $entry ( keys %$data )
	{
		foreach my $attr ( @$attrs )
		{
			$$data{ $entry }{ $attr } = $defaultEmptyString unless( defined
$$data{ $entry }{ $attr } );
		}
	}
}
```

```
sub setCardinality
{
	my ( $data, $attrs, $printAttrs ) = @_;
	my ( $count, $ref ) = ( undef ) x 2;
	my $dataCount = scalar keys %$data;

	foreach my $attr ( @$printAttrs )
	{
		$count = 0;

		foreach my $entry ( keys %$data )
		{
			$ref = \$$data{ $entry }{ $attr };

			if( ref $$ref eq 'ARRAY' && scalar @{ $$ref } > 1 )
			{
				$$attrs{ $attr }{ cardinal } = 'multiple';
				last;
			}
			else { $count += 1; }
		}

		$$attrs{ $attr }{ cardinal } = 'single' if( $count == $dataCount );
	}
}

sub printGEO
{
	my $ref = undef;

	open( my $fh, '>', "$softDir/$outputFile" );

	printGEOheader( $fh );

	foreach my $series ( sort keys %gse )
	{
		foreach my $sample ( @{ $gse{ $series }{ sample_id } } )
		{
			$ref = \$gsm{ $sample };

			next if( defined $targetOrganism && getAttrText( $$$ref{
organism_ch1 } ) ne $targetOrganism );

			foreach my $seriesAttr ( @printGSEattr )
			{
				print $fh ( getAttrText( $gse{ $series }{ $seriesAttr } ),
"\t" );
			}

			printGEOsamples( $ref, $fh );
		}
	}

	close $fh;
}
```

```
sub printGEOheader
{
	my $fh = $_[ 0 ];

	print $fh join( "\t", @printGSEattr );
	map { print $fh "\tsample_$_"; } @printGSMattr;
	print $fh "\n";
}

sub printGEOsamples
{
	my ( $ref, $fh ) = @_;
	my $sampleAttr = undef;

	for my $i ( 0 .. $#printGSMattr )
	{
		$sampleAttr = $printGSMattr[ $i ];

		print $fh getAttrText( $$$ref{ $sampleAttr } );

		if( $i == $#printGSMattr ) { print $fh "\n"; }
		else { print $fh "\t"; }
	}
}

sub printDataProcMethods
{
	my %methods;
	my ( $method, $date, $ref );
	my @header = qw( series_geo_accession pubmed_id submission_date
bioinformatics_method method_sample_IDs method_sample_count series_total_samples );

	open( my $fh, '>', "$softDir/$methodsOutfile" );

	print $fh ( join( "\t", @header ), "\n" );

	foreach my $series ( sort keys %gse )
	{
		%methods = ();
		$date = substr getAttrText( $gse{ $series }{ submission_date } ), -4;

		foreach my $sample ( @{ $gse{ $series }{ sample_id } } )
		{
			$method = getAttrText( $gsm{ $sample }{ data_processing } );
			$methods{ $method }{ count } += 1;
			push @{ $methods{ $method }{ samples } }, $sample;
		}
```

```
		foreach $method ( sort keys %methods )
		{
			$ref = $methods{ $method };

			print $fh ( "$series\t", getAttrText( $gse{ $series }{ pubmed_id
} ), "\t$date\t$method\t", getAttrText( $$ref{ samples } ), "\t$$ref{ count }\t$gse{
$series }{ totalOrganismCount }\n" );
		}
	}

	close $fh;
}

sub printCharTags
{
	return unless( scalar @printCHARattr > 0 );

	open( my $fh, '>', "$softDir/$tagsFile" );

	print $fh "series\tsample\ttag\tvalue\n";

	foreach my $sample ( sort keys %char )
	{
		foreach my $series ( @{ $gsm{ $sample }{ series_id } } )
		{
			map { print $fh "$series\t$sample\t$_\t$char{ $sample }{ $_ }\n";
} @printCHARattr;
		}
	}

	close $fh;
}

sub generateDB
{
	use DBI;

	my $dbh = DBI->connect( "dbi:SQLite:dbname=$softDir/$dbFile", undef, undef, {
AutoCommit => 0 } );

	generateDBtables( 'gse', $dbh, \%gse, \%gseAttr, \@printGSEattr );
	generateDBtables( 'gsm', $dbh, \%gsm, \%gsmAttr, \@printGSMattr );
	genCharTagsTable( 'char_tags', $dbh ) if( scalar @printCHARattr > 0 );

	$dbh->commit;
	$dbh->disconnect;
}
```

```
sub generateDBtables
{
	my ( $prefix, $dbh, $data, $attrs, $printAttrs ) = @_;
	my $ref = undef;
	my @singleAttrs = qw( geo_accession );
	my @pluralAttrs = ();

	foreach my $attr ( @$printAttrs )
	{
		$ref = \$$attrs{ $attr }{ cardinal };

		next if( $attr eq 'geo_accession' );
		next unless( defined $$ref );

		if( $$ref eq 'multiple' ) { push @pluralAttrs, $attr; }
		else { push @singleAttrs, $attr; }
	}

	createSingleTable( $prefix, $dbh, \@singleAttrs );
	createPluralTables( $prefix, $dbh, \@pluralAttrs );

	populateSingleTable( $prefix, $dbh, \@singleAttrs, $data );
	populatePluralTables( $prefix, $dbh, \@pluralAttrs, $data );
}

sub createSingleTable
{
	my ( $prefix, $dbh, $attrs ) = @_;
	my ( $sth, $query, $attr ) = ( undef ) x 3;

	$sth = $dbh->prepare( "DROP TABLE IF EXISTS $prefix;" );
	$sth->execute();

	$query = "CREATE TABLE $prefix ( ";

	for my $i ( 0 .. $#$attrs )
	{
		$attr = $$attrs[ $i ];

		if( $i == 0 ) { $query .= "`$attr` TEXT PRIMARY KEY, "; }
		elsif( $i == $#$attrs ) { $query .= "`$attr` TEXT );"; }
		else { $query .= "`$attr` TEXT, "; }
	}

	$sth = $dbh->prepare( $query );
	$sth->execute();
}
```

```
sub createPluralTables
{
	my ( $prefix, $dbh, $attrs ) = @_;
	my ( $tableName, $sth ) = ( undef ) x 2;

	foreach my $attr ( @$attrs )
	{
		$tableName = "${prefix}_${attr}";

		$sth = $dbh->prepare( "DROP TABLE IF EXISTS `$tableName`;" );
		$sth->execute();

		$sth = $dbh->prepare( "CREATE TABLE `$tableName` ( `geo_accession` TEXT,
`$attr` TEXT );" );
		$sth->execute();
	}
}

sub populateSingleTable
{
	my ( $prefix, $dbh, $attrs, $data ) = @_;
	my ( $sth, $insertAttrs, $insertPHs ) = ( undef ) x 3;

	$insertAttrs = join( ', ', map { "`$_`" } @$attrs );
	$insertPHs = join( ', ', map { '?' } @$attrs );
	$sth = $dbh->prepare( "INSERT INTO $prefix ( $insertAttrs ) VALUES (
$insertPHs );" );

	foreach my $entry ( sort keys %$data )
	{
		$sth->execute( map { getAttrText( $$data{ $entry }{ $_ } ); } @$attrs );
	}
}
```

```
sub populatePluralTables
{
	my ( $prefix, $dbh, $attrs, $data ) = @_;
	my ( $tableName, $sth, $ref ) = ( undef ) x 3;

	foreach my $attr ( @$attrs )
	{
		$tableName = "${prefix}_${attr}";
		$sth = $dbh->prepare( "INSERT INTO `$tableName` ( `geo_accession`,
`$attr` ) VALUES ( ?, ? );" );

		foreach my $entry ( sort keys %$data )
		{
			$ref = \$$data{ $entry }{ $attr };

			if( ref $$ref eq 'ARRAY' )
			{
				map { $sth->execute( $entry, $_ ); } @{ $$ref };
			}
			else { $sth->execute( $entry, $$ref ); }
		}
	}
}

sub genCharTagsTable
{
	my ( $prefix, $dbh ) = @_;
	my ( $sth, $createAttrs, $insertAttrs, $insertPHs ) = ( undef ) x 4;
	my @attrs = qw( series sample tag value );

	$sth = $dbh->prepare( "DROP TABLE IF EXISTS $prefix;" );
	$sth->execute();

	$createAttrs = join( ', ', map { "`$_` TEXT" } @attrs );
	$sth = $dbh->prepare( "CREATE TABLE $prefix ( $createAttrs );" );
	$sth->execute();

	$insertAttrs = join( ', ', map { "`$_`" } @attrs );
	$insertPHs = join( ', ', map { '?' } @attrs );
	$sth = $dbh->prepare( "INSERT INTO $prefix ( $insertAttrs ) VALUES (
$insertPHs );" );

	foreach my $sample ( sort keys %char )
	{
		foreach my $series ( @{ $gsm{ $sample }{ series_id } } )
		{
			map { $sth->execute( $series, $sample, $_, $char{ $sample }{ $_ }
); } @printCHARattr;
		}
	}
}
```

```
sub main
{
	if( scalar @ARGV == 0 ) { printHelp(); }
	else { parseArgs(); }
}

main();
```

geomp_pmc_pull

```
#!/usr/bin/env bash
# Aurash

if [[ $# -eq 0 || ${1} == "-h" || ${1} == "--help" ]]
then
	printf "\nUsage: [./]geomp_pmc_pull softDirectory/methodsFile\n\n"
	exit
fi

methodsFile=${1}
methodsDir=${methodsFile%/*}
dirlessMethodsFile=${methodsFile##*/}

if [[ ! -d ${methodsDir} ]]
then
	printf "\n\"%s\" is not a valid directory!\n\n" ${methodsDir}
	exit
fi

if [[ ! -f ${methodsDir}/${dirlessMethodsFile} ]]
then
	printf "\n\"%s\" is not a valid file!\n\n" ${dirlessMethodsFile}
	exit
fi

# Sanity check
# printf "\nmethodsFile: ${methodsFile}\nmethodsDir:
${methodsDir}\ndirlessMethodsFile: ${dirlessMethodsFile}\n\n"
# exit

pushd ${methodsDir} > /dev/null

# Create the pdf and tarball
# directories if they do not
# exist
for dir in $(printf "pdf tarball")
do
	if [[ ! -d ${dir} ]]
	then
		mkdir ${dir}
		printf "Directory generated: ${dir}\n"
	fi
done

column=$(head -n 1 ${dirlessMethodsFile} | sed 's/\t/\n/g' | awk 'BEGIN { count = 0;
} { count += 1; if( $0 == "pubmed_id" ) { print count; exit; } }')
```

```
for pubmedID in $(tail -n +2 ${dirlessMethodsFile} | awk -v column=${column} 'BEGIN {
FS="\t"; } { print $column; }' | grep "^[0-9]\+" | sed 's/||/\n/g' | sort -u)
do
	# ID conversion from pubmedIDs
	# to pmcIDs
	printf "Attempting ID conversion for pubmedID ${pubmedID} ... "
	pmcID=$(wget -q -O -
"https://www.ncbi.nlm.nih.gov/pmc/utils/idconv/v1.0/?ids=${pubmedID}&tool=geomp_pmc_p
ull&email=aurash@uwm.edu" | grep -o "pmcid=\"[^\"]\+" | head -n 1 | sed
's/^pmcid="//')
	printf 'done!\n'
	sleep 2

	if [[ $(printf "${pmcID}" | wc -c) -lt 4 ]]
	then
		printf "SKIPPING pubmedID ${pubmedID}, no relevant pmcID found ...\n\n"
		continue
	fi

	# FTP link acquisition for
	# PDFs attached to pmcIDs
	printf "Attempting to fetch FTP link(s) for pubmedID ${pubmedID} ... "
	links=$(wget -q -O -
"https://www.ncbi.nlm.nih.gov/pmc/utils/oa/oa.fcgi?id=${pmcID}&tool=geomp_pmc_pull&em
ail=aurash@uwm.edu" | grep -o "href=\"[^\"]\+" | sed 's/^href="//')
	printf 'done!\n'
	sleep 2

	if [[ $(printf "${links}" | wc -c) -lt 11 ]]
	then
		printf "SKIPPING pubmedID ${pubmedID}, no FTP links found...\n\n"
		continue
	fi

	# Sanity check
	# printf "${links}\n"

	# Create pubmedID directory
	# if it does not exist
	if [[ ! -d pdf/${pubmedID} ]]
	then
		mkdir pdf/${pubmedID}
		printf "Directory generated: pdf/${pubmedID}\n"
	fi

	# Post-processing of FTP links
	# to actually download the
	# desired archives / files
	pdfLink=$(printf "${links}" | grep "\.pdf$")
	pdfLinkLength=$(printf "${pdfLink}" | wc -c)
	tarLink=$(printf "${links}" | grep "\.tar\.gz$")
	tarLinkLength=$(printf "${tarLink}" | wc -c)
```

```
	if [[ ${pdfLinkLength} -gt 10 ]]
	then
		printf "Downloading PDF file directly ... "
		wget -q -O pdf/${pubmedID}/${pubmedID}.pdf "${pdfLink}"
		printf 'done!\n'
	fi

	if [[ ${tarLinkLength} -gt 10 ]]
	then
		printf "Downloading *.tar.gz archive ... "
		wget -q -O ${pubmedID}.tar.gz "${tarLink}"
		printf 'done!\n'

		for file in $(tar -tzf ${pubmedID}.tar.gz --wildcards "*.pdf")
		do
			name=${file##*/}
			numComponents=$(printf "${file}" | grep -o "/" | wc -l)

			printf "Extracting ${name} to pdf/${pubmedID}/ ... "
			tar -xzf ${pubmedID}.tar.gz "${file}" --strip-
components=${numComponents}
			mv ${name} pdf/${pubmedID}
			printf 'done!\n'
		done

		printf "Moving ${pubmedID}.tar.gz ... "
		mv ${pubmedID}.tar.gz tarball
		printf 'done!\n'
	else
		printf "ERROR, failure to process:\n${links}\n"
	fi

	echo
	sleep 2
done

popd > /dev/null
```

fixGSElist

```
#!/usr/bin/env perl
# Aurash

use strict;
use warnings;
use autodie;

my ( $gse, $pubmed ) = ( undef ) x 2;

open( my $fh, '<', 'gse.list' );

while( my $line = <$fh> )
{
	( $gse, $pubmed ) = split /\t/, $line;

	if( $pubmed =~ m/\|\|/ )
	{
		chomp $pubmed;

		map { print "$gse\t$_\n"; } split( /\|\|/, $pubmed );
	}
	else { print $line; }
}

close $fh;
```

genGSEtitles

```
#!/usr/bin/env bash
# Aurash

gseList=${1}

if [[ ${gseList} == "" ]]
then
	printf "\nERROR: no GSE list file provided\n\n"
	exit
fi

printf "GSE_accession_ID\tpubmed_ID\tpublication_title\n"

while IFS=$'\t' read gseID pubmedID
do
	# Sanity check
	# printf "${gseID}\t${pubmedID}\n"

	pubmedFile="${pubmedID}.efetch"

	if [[ -f ${pubmedFile} ]]
	then
		printf "${pubmedFile} already exists\n"
	else
		printf "Pulling ${pubmedFile} ... "
		wget -q -O ${pubmedFile}
"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&id=${pubmedID}"
		printf "done\n"
	fi

	title=$(./slurpEFetch ${pubmedFile})

	printf "${gseID}\t${pubmedID}\t${title}\n"
done < ${gseList}
```

slurpEFetch

```
#!/usr/bin/env perl
# Aurash

use strict;
use warnings;
use autodie;

my $file = $ARGV[ 0 ] || undef;

die "Error: no file provided" unless( defined $file );
die "Error: input file does not appear to be an EFetch result" unless( $file =~
m/\.efetch$/i );

open( my $fh, '<', $file );
read( $fh, my $text, -s $file );
close $fh;

if( $text =~ m/^\s*title \{\s*\n\s*name "([^"]+)"/ms )
{
	my $title = $1; $title =~ s/[.\n]//g;
	print "$title\n";
}
else { print "Error: no title found for $file\n"; exit; }
```

georget_pull

```
#!/usr/bin/env perl
# Aurash

use strict;
use warnings;
use autodie;
use DBI;
use File::Copy 'mv';
use Net::FTP;

my ( $dbLoc, $dbFile, $dbDir, $email ) = ( undef ) x 4;

my %opts =
(
	-i => { var => \$dbLoc, default => undef },
	-e => { var => \$email, default => undef }
);

sub valid
{
	my $value = $_[ 0 ];

	if( defined $value && $value ne '' ) { return 1; }
	else { return 0; }
}

sub parseArgs
{
	for my $i ( 0 .. $#ARGV )
	{
		parseOpt( $i ) if( substr( $ARGV[ $i ], 0, 1 ) eq '-' );
	}

	# Sanity check
	# print "\$dbLoc: $dbLoc\n\$dbDir: $dbDir\n\$dbFile: $dbFile\n\$email:
$email\n";

	die "No database location provided" unless( valid( $dbLoc ) );
	die "Database location appears invalid" unless( $dbLoc =~ m/^[^\/]+\/[^\/]+$/
);
	die "No email address provided" unless( valid( $email ) );
	die "Email address appears invalid" unless( $email =~ m/^[^@]+@[^@]+$/ );

	( $dbFile = $dbLoc ) =~ s/^.*\///;
	( $dbDir = $dbLoc ) =~ s/\/[^\/]*$//;

	prepDirs();
}
```

```
sub parseOpt
{
	my ( $i, $opt, $value ) = ( $_[ 0 ], $ARGV[ $_[ 0 ] ], undef );
	$value = $ARGV[ $i + 1 ] if( defined $ARGV[ $i + 1 ] && substr( $ARGV[ $i + 1
], 0, 1 ) ne '-' );

	if( defined $opts{ $opt } )
	{
		if( defined $value )
		{
			${ $opts{ $opt }{ var } } = $value;
		}
		else
		{
			${ $opts{ $opt }{ var } } = $opts{ $opt }{ default };
		}
	}
	elsif( $opt eq '-h' || $opt eq '--help' ) { printHelp(); exit; }
	else { die "Unrecognized option: '$opt'"; }
}

sub printHelp
{
	print << 'EOF';

GEORGET_PULL: a Supplemental Analysis Extractor for the Gene Expression Omnibus

	by Aurash Mohaimani

USAGE:
	[./]georget_pull -i databaseLocation -e emailAddress

OPTIONS:
	-i databaseLocation
		REQUIRED: specifies the path to the SQLite database to be used
		in pulling supplemental analyses from GEO within the NCBI FTP
		server. This path must be of the form "directory/dbFile".

	-e emailAddress
		REQUIRED: defines the user email address to be used upon login
		to the NCBI FTP server.

	-h, --help
		Prints this help message.

EOF
}
```

```
sub prepDirs
{
	chdir $dbDir;

	foreach my $dir ( qw( GSE_supp GSM_supp ) )
	{
		mkdir $dir, 0774 unless( -d $dir );
	}

	accessDB();
}

sub accessDB
{
	my $dbh = DBI->connect( "dbi:SQLite:dbname=${dbFile}", '', '' );
	my $ftp = Net::FTP->new( 'ftp.ncbi.nlm.nih.gov', Debug => 0 );
	$ftp->login( 'anonymous', $email );
	$ftp->binary;

	accessDBtable( $dbh, 'gse_supplementary_file', 'GSE_supp', $ftp );
	accessDBtable( $dbh, 'gsm_supplementary_file', 'GSM_supp', $ftp );
}
```

```
sub accessDBtable
{
	my ( $dbh, $table, $baseDir, $ftp ) = @_;
	my ( $id, $url, $file, $dir, $ftpPath, $fileSize ) = ( undef ) x 6;
	my $sth = $dbh->prepare( "SELECT * FROM $table;" );

	$sth->execute();

	while( ( $id, $url ) = $sth->fetchrow_array )
	{
		next unless( $url =~ m/^ftp:\/\// );
		next if( $url =~ m/\/sra\/sra-instant\/|_RAW\.tar$/ );

		( $file = $url ) =~ s/^.*\///;
		$dir = "${baseDir}/${id}";
		( $ftpPath = $url ) =~ s/^ftp:\/\/ftp\.ncbi\.nlm\.nih\.gov//;
		$fileSize = $ftp->size( $ftpPath ) / 1000000;

		next if( $fileSize > 250 );

		# Sanity check
		# print "$id\t$file\t$fileSize\n"; next;

		mkdir $dir, 0774 unless( -d $dir );

		if( -e "${dir}/${file}" ) { print "$file exists\n"; }
		else
		{
			print "Pulling $file ... ";
			$ftp->get( $ftpPath );
			mv $file, $dir;
			print "done\n";
		}
	}
}

sub main
{
	if( scalar @ARGV == 0 ) { printHelp(); }
	else { parseArgs(); }
}

main();
```

georget_cull

```
#!/usr/bin/env bash
# Aurash

if [[ $# -eq 0 || ${1} == '-h' || ${1} == '--help' ]]
then
	printf "\nUsage: [./]georget_cull blacklistFile directory\n\n"
	exit
fi

blacklist=${1}
mainDir=${2}

if [[ ! -f ${blacklist} ]]
then
	printf '\nNo blacklist file provided!\n\n'
	exit
elif [[ ${mainDir} == "" ]]
then
	printf '\nNo directory provided!\n\n'
	exit
elif [[ ! -d ${mainDir} ]]
then
	printf "\n${mainDir} is not a valid directory\n\n"
	exit
fi

pushd ${mainDir} > /dev/null

while read accID
do
	GSEdir="GSE_supp/${accID}"

	if [[ -d ${GSEdir} ]]
	then
		rm -r ${GSEdir}
		printf "${GSEdir} removed\n"
	fi

	for dir in $(grep "^^SAMPLE" ${accID}.soft | sed
's/^^SAMPLE\s\+=\s\+/GSM_supp\//; s/\s*$//')
	do
		if [[ -d ${dir} ]]
		then
			rm -r ${dir}
			printf "${dir} removed\n"
		fi
	done
done < ../${blacklist}

popd > /dev/null
```

georget_decompress

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

for dir in $(printf "GSE_supp GSM_supp")
do
	if [[ ! -d ${dir} ]]
	then
		printf "\n${dir} directory not found"
		printf '!\n\n'
		exit
	fi
done

for file in $(find . -mindepth 3 -type f -name "*.gz")
do
	printf "Unzipping ${file} ... "
	gzip -d ${file}
	printf "done\n"
done

popd > /dev/null
```

georget_decompress_parallel

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

for dir in $(printf "GSE_supp GSM_supp")
do
	if [[ ! -d ${dir} ]]
	then
		printf "\n${dir} directory not found"
		printf '!\n\n'
		exit
	fi
done

find . -mindepth 3 -type f -name "*.gz" | parallel 'gzip -d {}; printf "Unzipped
{.}\n";'

popd > /dev/null
```

georget_truncate

```
#!/usr/bin/env bash
# Aurash

if [[ "$#" -lt 2 || ${1} == "-h" || ${1} == "--help" ]]
then
	printf "\nUsage: [./]georget_truncate directory fileSizeLimit\n\n"
	exit
fi

dir=${1}
limit=${2}

if [[ ! -d ${dir} ]]
then
	printf "\nERROR: ${dir} is not a directory"
	printf '!\n\n'
	exit
elif [[ ! ${limit} =~ ^[0-9]+$ ]]
then
	printf "\nERROR: fileSizeLimit must be a positive integer\n\n"
	exit
fi

pushd ${dir} > /dev/null

for file in $(du -B 1000000 $(find . -mindepth 3 -type f) | awk -v limit="${limit}"
'{ if( $1 > limit ) { print $2; } }')
do
	rm ${file}
	printf "Removed: ${file}\n"
done

popd > /dev/null
```

georget_reformat

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

for file in $(find . -mindepth 3 -type f)
do
	type=$(file -b ${file})

	if [[ ${type} =~ "with CR line" ]]
	then
		mac2unix ${file}
	fi
done

popd > /dev/null
```

georget_sanitize

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

for file in $(find . -mindepth 3 -type f | grep -i "\.csv$\|\.tsv$\|\.txt$")
do
	sed 's/^\s*\|\s*$//g' ${file} > ${file}.tmp
	mv ${file}.tmp ${file}
	printf "Sanitized ${file}\n"
done

popd > /dev/null
```

georget_sanitize_parallel

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

find . -mindepth 3 -type f | grep -i "\.csv$\|\.tsv$\|\.txt$" | parallel 'sed
"s/^\s*\|\s*$//g" {} > {}.tmp; mv {}.tmp {}; printf "Sanitized {}\n";'

popd > /dev/null
```

georget_sanitize_MACS

```
#!/usr/bin/env bash
# Aurash

dir=${1}

if [[ ${dir} == "" ]]
then
	printf '\nNo input directory specified!\n\n'
	exit
fi

pushd ${dir} > /dev/null

for file in $(find . -mindepth 3 -type f | grep -i "\.txt$")
do
	match=$(head -n 5 ${file} | grep "generated by MACS version")

	if [[ ${match} =~ "generated by MACS version" ]]
	then
		sed '/^\s*#\|^\s*$/d' ${file} > ${file}.tmp
		mv ${file}.tmp ${file}
		printf "Sanitized MACS file: ${file}\n"
	fi
done

popd > /dev/null
```

georget

```
#!/usr/bin/env perl
# Aurash

use strict;
use warnings;
use autodie;
use DBI;

my ( $fileSizeLimit, $ambiguityLimit, $genIndexes ) = ( 500, 95, undef );
my ( $rawTotal, $adjTotal, $dbTotal ) = ( 0 ) x 3;
my ( $dbh, $sth, $sthGFF, $dbLoc, $dbDir, $dbFile ) = ( undef ) x 6;
my %opts =
(
	-i => { var => \$dbLoc, default => undef },
	-s => { var => \$fileSizeLimit, default => 500 },
	-a => { var => \$ambiguityLimit, default => 95 },
	-x => { var => \$genIndexes, default => 2 }
);

sub parseArgs
{
	for my $i ( 0 .. $#ARGV )
	{
		parseOpt( $i ) if( defined $ARGV[ $i ] && substr( $ARGV[ $i ], 0, 1 ) eq
'-' );
	}

	die "\$dbLoc is undefined" unless( defined $dbLoc );
	die "\$dbLoc does not appear valid" unless( $dbLoc =~ m/^[^\/]+\/[^\/]+\.db$/
);
	die "\$fileSizeLimit must be a positive integer" unless( $fileSizeLimit =~
m/^[0-9]+$/ );
	die "\$ambiguityLimit must be a positive integer" unless( $ambiguityLimit =~
m/^[0-9]+$/ );
	die "\$ambiguityLimit out of bounds" unless( $ambiguityLimit > 0 &&
$ambiguityLimit < 100 );
	die "\$genIndexes out of bounds" if( defined $genIndexes && $genIndexes !~
m/^[012]$/ );

	$dbDir = $dbLoc; $dbDir =~ s/\/[^\/]+$//;
	$dbFile = $dbLoc; $dbFile =~ s/^.*\///;

	chdir $dbDir;

	$dbh = DBI->connect( "dbi:SQLite:dbname=$dbFile", '', '', { AutoCommit => 0 }
);
	$dbh->do( "PRAGMA temp_store = $genIndexes;" );

	map { scanDirectory( $_ ); } qw( GSE_supp GSM_supp );
```

```
	$dbh->commit;
	$dbh->disconnect;

	printMetrics();
}

sub parseOpt
{
	my ( $i, $opt, $val ) = ( $_[ 0 ], $ARGV[ $_[ 0 ] ], undef );
	$val = $ARGV[ $i + 1 ] if( defined $ARGV[ $i + 1 ] && substr( $ARGV[ $i + 1 ],
0, 1 ) ne '-' );

	if( defined $opts{ $opt } )
	{
		${ $opts{ $opt }{ var } } = defined $val ? $val : $opts{ $opt }{ default
};
	}
	elsif( $opt eq '-h' || $opt eq '--help' ) { printHelp(); exit; }
	else { die "Unrecognized option: '${opt}'"; }
}

sub printHelp
{
	print << 'EOF';

GEORGET: an RNA-seq Gene Expression Results Translator for the
	 Gene Expression Omnibus

	by Aurash Mohaimani, dedicated to Purrito & B.B.

USAGE:
	[./]georget -i databaseLocation [-s[ fileSizeLimit]]
	[-a[ ambiguityLimit]] [-x[ tempStore]]

OPTIONS:
	-i databaseLocation
		REQUIRED: specifies the path to the SQLite database to be used
		when translating and loading RNA-seq results that have been
		previously pulled from GEO. This path must be of the form
		"directory/dbFile".

	-s[ fileSizeLimit]
		Defines the maximum file size, in megabytes, for results files
		that will be translated and loaded into the database. The
		provided value must be a positive integer. (default: 500)

	-a[ ambiguityLimit]
		Specifies the minimum threshold at which an ambiguous file
		(e.g. a heterogeneous file with a non-uniform number of
		columns) will be processed for translation. This value must be
		a positive integer less than 100. (default: 95)
```

```
	-x[ tempStore]
		Instructs GEORGET to construct database indexes for each table
		of translated RNA-seq results. This flag can be specified with
		an explicit PRAGMA temp_store value of 0, 1, or 2.
		(default: disabled or 2 when ambiguously declared)

	-h, --help
		Prints this help message.

EOF
}

sub scanDirectory
{
	my ( $dir, $ext ) = ( $_[ 0 ], undef );
	my $gff = $dir . '_GXF';

	$dbh->do( "DROP TABLE IF EXISTS $dir;" );
	$dbh->do( "DROP TABLE IF EXISTS $gff;" );

	$dbh->do( "CREATE TABLE $dir ( accessionID TEXT, file TEXT, feature_name TEXT,
feature_value TEXT, attr_name TEXT, attr_value TEXT );" );
	$dbh->do( "CREATE TABLE $gff ( accessionID TEXT, file TEXT, seqID TEXT, source
TEXT, type TEXT, start TEXT, end TEXT, score TEXT, strand TEXT, phase TEXT,
attributes TEXT );" );

	$sth = $dbh->prepare( "INSERT INTO $dir ( accessionID, file, feature_name,
feature_value, attr_name, attr_value ) VALUES ( ?, ?, ?, ?, ?, ? );" );
	$sthGFF = $dbh->prepare( "INSERT INTO $gff ( accessionID, file, seqID, source,
type, start, end, score, strand, phase, attributes ) VALUES ( ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ? );" );

	chomp( my @files = qx{ find $dir -type f } );
	@files = grep { $_ !~ m/readme/i; } @files;
	$rawTotal += scalar @files;

	foreach my $file ( @files )
	{
		$ext = $file; $ext =~ s/^.*\.//;

		if( $file =~ m/intensit/i || $ext =~
m/^bed|^bg|^wig|^big|^bw|^bam|^fn?a|^vcf|^xls|^xml|^pdf|^json/i )
		{
			print "[WARNING] $file :: incompatible format\n\n";
		}
		else { $adjTotal += 1; processFile( $file ); }
	}
```

```
	if( defined $genIndexes )
	{
		$dbh->do( "CREATE INDEX idx_${dir} ON $dir ( accessionID, file,
feature_value, attr_name );" );
		print "[IDX] $dir :: database table index generated\n\n";

		$dbh->do( "CREATE INDEX idx_${gff} ON $gff ( accessionID, file, seqID,
start, end );" );
		print "[IDX] $gff :: database table index generated\n\n";
	}
}

sub processFile
{
	my $file = $_[ 0 ];
	my $size = -s $file; $size = int( $size / 1000000 );

	if( $size >= $fileSizeLimit )
	{
		print "[WARNING] $file :: file size ($size MB) too large\n\n";
		return;
	}

	my $sep = findSeparator( $file ); $sep = qr/$sep/;
	my ( $lineCount, $numFields ) = ( 0, undef );
	my %fields = ();

	open( my $fh, '<', $file );

	while( my $line = <$fh> )
	{
		$lineCount += 1;
		$numFields = split /$sep/, $line;
		$fields{ $numFields }{ count } += 1;
	}

	close $fh;
```

```
	if( scalar keys %fields > 1 )
	{
		disambiguateFile( $file, $sep, $lineCount, \%fields );
	}
	else
	{
		my ( $columnCount ) = keys %fields;

		if( $columnCount == 1 )
		{
			print "[WARNING] $file :: \@attrs == 1\n\n";
			return;
		}
		elsif( $columnCount == 9 && isGFF( $file ) )
		{
			print "[GOOD] $file :: is a GFF\n";
			insertGFFfile( $file );
		}
		else
		{
			print "[GOOD] $file :: appears uniform\n";
			insertUniformFile( $file, $sep );
		}
	}
}

sub findSeparator
{
	my ( $file, $lineCount ) = ( $_[ 0 ], 0 );
	my %seps = ( "\t" => 0, ',' => 0 );

	open( my $fh, '<', $file );

	while( my $line = <$fh> )
	{
		$lineCount += 1;

		foreach my $sep ( keys %seps )
		{
			$seps{ $sep } += split "$sep", $line;
		}

		last if( $lineCount == 15 );
	}

	close $fh;

	foreach my $sep ( sort { $seps{ $b } <=> $seps{ $a } } keys %seps )
	{
		return $sep;
	}
}
```

```
sub disambiguateFile
{
	my ( $file, $sep, $lineCount, $fields ) = @_;
	my ( $columnCount, $numFields, $numAttrs ) = ( undef ) x 3;
	my @attrs = ();

	map { $$fields{ $_ }{ perc } = $$fields{ $_ }{ count } / $lineCount * 100; }
keys %$fields;

	foreach my $field ( sort { $$fields{ $b }{ count } <=> $$fields{ $a }{ count }
} keys %$fields )
	{
		$columnCount = $field if( $$fields{ $field }{ perc } >= $ambiguityLimit
);
		last;
	}

	unless( defined $columnCount )
	{
		print "[WARNING] $file :: unable to determine \$columnCount\n";
		printColumnFrequency( $fields );
		return;
	}

	if( $columnCount == 9 && isGFF( $file ) )
	{
		print "[GOOD] $file :: is a GFF\n";
		insertGFFfile( $file );
		return;
	}

	open( my $fh, '<', $file );

	while( my $line = <$fh> )
	{
		$numFields = split /$sep/, $line;

		next if( $numFields < $columnCount - 1 );

		chomp $line;
		@attrs = split /$sep/, $line;
		$numAttrs = @attrs;
		last;
	}

	if( $numAttrs == 1 )
	{
		printFileReport( "[WARNING] $file :: \@attrs == 1\n", \@attrs, $fields
);
		return;
	}
```

```
	if( $numAttrs > $columnCount )
	{
		printFileReport( "[WARNING] $file :: \$numAttrs > \$columnCount, \@attrs
could not be resolved\n", \@attrs, $fields );
		return;
	}
	elsif( $columnCount - $numAttrs == 1 )
	{
		if( $attrs[ 0 ] =~ m/gene|transcript|refseq|ensembl|id/i )
		{
			printFileReport( "[WARNING] $file :: \$columnCount - \$numAttrs
== 1 and \@attrs could not be resolved\n", \@attrs, $fields );
			return;
		}
		else { unshift @attrs, 'featureID'; $numAttrs = @attrs; }
	}
	elsif( $columnCount - $numAttrs > 1 )
	{
		printFileReport( "[WARNING] $file :: \$columnCount - \$numAttrs > 1,
\@attrs could not be resolved\n", \@attrs, $fields );
		return;
	}

	# Prune @attrs of banned characters
	map { $_ =~ s/["']//g; } @attrs;

	printFileReport( "[OK] $file ::\n-------------\n\$columnCount:
$columnCount\n", \@attrs, $fields );
	insertAmbiguousFile( $file, $sep, $fh, $columnCount, \@attrs );
}
```

```
sub isGFF
{
	my $file = $_[ 0 ];
	my ( $totalLineCount, $validLineCount, $validPerc ) = ( ( 0 ) x 2, undef );
	my ( $seqID, $source, $type, $start, $end, $score, $strand, $phase ) = ( undef
) x 8;

	open( my $fh, '<', $file );

	while( my $line = <$fh> )
	{
		next if( $line =~ m/^\s*#/ );

		$totalLineCount += 1;
		( $seqID, $source, $type, $start, $end, $score, $strand, $phase ) =
split /\t/, $line;

		next unless( $seqID =~ m/^[[:alnum:].:\^*\$@!+_?\-|]+$/ );
		next unless( $type =~ m/^[[:alpha:]]+$|^[Ss][Oo]:[0-9]+$/ );
		next unless( $start =~ m/^[0-9]+$/ && $end =~ m/^[0-9]+$/ );
		next unless( $score =~ m/^[0-9]+(\.[0-9]*)?([eE]{1,2}[+-]?[0-
9]+)?$|^\.$/ );
		next unless( $strand =~ m/^[.+?-]$/ );
		next unless( $phase =~ m/^[012.]$/ );

		$validLineCount += 1;

		last if( $totalLineCount == 10000 );
	}

	close $fh;

	$validPerc = $validLineCount / $totalLineCount * 100;

	if( $validPerc >= 95 ) { return 1; }
	else { return 0; }
}

sub printFileReport
{
	my ( $string, $attrs, $fields ) = @_;

	print $string;
	printColumns( $attrs );
	printColumnFrequency( $fields );
}

sub printColumns
{
	my ( $attrs, $bars ) = ( $_[ 0 ], '-------------' );

	print ( "$bars\nColumns (", scalar( @$attrs ), "):\n$bars\n", join( "\n",
@$attrs ), "\n" );
}
```

```
sub printColumnFrequency
{
	my ( $fields, $bars ) = ( $_[ 0 ], '-----------------' );

	print "$bars\nColumn frequency:\n$bars\n";
	map { print "$_\t$$fields{ $_ }{ count }\t$$fields{ $_ }{ perc }\n"; } sort {
$b <=> $a } keys %$fields;
	print "\n";
}

sub insertGFFfile
{
	my $file = $_[ 0 ];
	my @data = ();

	open( my $fh, '<', $file );

	$file =~ s/^.*\///;
	my ( $accID, $baseName ) = split /[_.]/, $file, 2;

	while( my $line = <$fh> )
	{
		next if( $line =~ m/^\s*#/ );

		chomp $line;
		@data = split /\t/, $line;

		next unless( scalar( @data ) == 9 );

		$sthGFF->execute( $accID, $baseName, @data );
	}

	close $fh; $dbTotal += 1;
	print "[DB] $file :: successfully inserted\n\n";
}
```

```
sub insertUniformFile
{
	my ( $file, $sep ) = @_;
	my $lineCount = 0;
	my ( @attrs, @data ) = ();

	open( my $fh, '<', $file );

	$file =~ s/^.*\///;
	my ( $accID, $baseName ) = split /[_.]/, $file, 2;

	while( my $line = <$fh> )
	{
		$lineCount += 1;
		chomp $line;

		if( $lineCount == 1 ) { @attrs = split /$sep/, $line; }
		else
		{
			@data = split /$sep/, $line;

			map { $sth->execute( $accID, $baseName, $attrs[ 0 ], $data[ 0 ],
$attrs[ $_ ], $data[ $_ ] ); } ( 1 .. $#attrs );
		}
	}

	close $fh; $dbTotal += 1;
	print "[DB] $file :: successfully inserted\n\n";
}

sub insertAmbiguousFile
{
	my ( $file, $sep, $fh, $columnCount, $attrs ) = @_;
	$file =~ s/^.*\///;
	my ( $accID, $baseName ) = split /[_.]/, $file, 2;
	my @data = ();

	while( my $line = <$fh> )
	{
		chomp $line;
		@data = split /$sep/, $line;

		next unless( $columnCount == scalar( @data ) );

		map { $sth->execute( $accID, $baseName, $$attrs[ 0 ], $data[ 0 ],
$$attrs[ $_ ], $data[ $_ ] ); } ( 1 .. $#$attrs );
	}

	close $fh; $dbTotal += 1;
	print "[DB] $file :: successfully inserted\n\n";
}
```

```
sub printMetrics
{
	my $rawTransRate = sprintf '%.2f%%', $dbTotal / $rawTotal * 100;
	my $adjTransRate = sprintf '%.2f%%', $dbTotal / $adjTotal * 100;

	print "Raw translation rate: $dbTotal / $rawTotal files
(${rawTransRate})\nAdjusted translation rate: $dbTotal / $adjTotal files
(${adjTransRate})\n\n";
}

sub main
{
	scalar( @ARGV ) == 0 ? printHelp() : parseArgs();
}

main();
```

CURRICULUM VITAE

Aurash Mohaimani

Education History

Sep 2012 – May 2018	Ph.D. in Biomedical and Health Informatics, University of Wisconsin-Milwaukee Dissertation Topic: GEORAC, an RNA-Seq Atlas Constructor for the Gene Expression Omnibus
Aug 2007 – May 2011	B.S. in Molecular Biochemistry and Biophysics (Magna Cum Laude), Illinois Institute of Technology (GPA: 3.8)
Aug 2004 – Jun 2007	Illinois Mathematics and Science Academy

Academic Research Experience

Oct 2017 – Present	Bioinformatics: *de novo* Whole Genome Assembly and Annotation of Multiple Aquatic Species via the Fund for Lake Michigan (FFLM)
Oct 2016 – Jun 2017	Bioinformatics: Re-development of an Ensembl BioMart-based Ortholog Mapper and Genome Sequence Extractor for CPAT-facilitated lncRNA identification of Zebrafish RNA-Seq Data
Apr 2016 – Apr 2017	Bioinformatics: Execution and Subsequent Design of an Annotation Pipeline for Barcoded IsoSeq Results of Daphnia
Feb 2016 – Present	Bioinformatics: Porting, Re-implementation, and Refinement of a Global Alignment Sequence Taxonomy (GAST) Workflow for Microbial 16S Datasets
Feb 2016 – Aug 2017	Bioinformatics: Implementation and Refinement of a Command-line Pipeline for 16S Dataset Results Generated via Illumina Reporter
May 2015 – Present	Bioinformatics: Development and Prototypical Implementation of GEORAC, an RNA-seq Atlas Construction Workflow for the Gene Expression Omnibus (GEO)
Feb 2014 – Feb 2016	Bioinformatics: Installation, Migration, and Refinement of a next-generation asynchronous JavaScript-based genome browser supporting Whole Genome Sequencing (WGS) datasets
Apr 2013 – Aug 2013	Bioinformatics: *de novo* RNA-Seq Assembly and Analysis of *Montastraea faveolata*
Jan 2013 – May 2013	Public Health: Assessment of Breast Cancer Risk in Morocco
Oct 2012 – Aug 2013	Bioinformatics: A Zebrafish Cloud Computational Resource for Environmental Health Science Research
Sep 2012 – Jun 2013	Limnology: Actionable Public Health Information from a Temporal Milwaukee Beach Water Quality Database
May 2010 – Dec 2010	Biophysics: Performance of Molecular Dynamics Simulations via GROMACS
Jul 2008 – Jun 2009	Physiology: Literature Review of Image Distance Estimation during Eye Saccades
Sep 2005 – May 2006	Molecular Biology: Injection of Bone Marrow-Derived Stem Cells into Dystrophic Mice

Academic Courses Taught

Health Sciences 105	Survey of the Health Professions
Health Sciences 250	Allied Health Information Methods (EHRs / EMRs)
Health Care Admin 460	Healthcare Reimbursement Systems

Work and Volunteering Experience

Feb 2016 – Present	Bioinformatics Research Specialist within the Great Lakes Genomics Center (GLGC) at the School of Freshwater Sciences (SFS), University of Wisconsin-Milwaukee (UW-Milwaukee)
Sep 2014 – May 2015	Adjunct Instructor within the College of Health Sciences (CHS) at the University of Wisconsin-Milwaukee (UW-Milwaukee)
Feb 2014 – Feb 2016	Bioinformatics Data Specialist within the Rat Genome Database (RGD) at the Human and Molecular Genetics Center (HMGC), Medical College of Wisconsin (MCW)
Sep 2012 – Aug 2013	Project Assistant in the Laboratory for Public Health Informatics and Genomics (LPHIG) at the University of Wisconsin-Milwaukee (UW-Milwaukee)
Jun 2011 – Jul 2011	Laboratory technician with IMCA-CAT at the Advanced Photon Source (APS) at Argonne National Laboratory
May 2008 – Jun 2009	E.R. volunteer at South Suburban Hospital in Hazelcrest, Illinois
Aug 2008 – Dec 2008	Mathematics Teaching Assistant at the Illinois Institute of Technology

Conferences and Workshops Attended

Sep 26th – 27th 2016	The Second Annual Third Coast Biomedical and Health Informatics Consortium Conference: Big Data Workshop (Oral Presentation: A Massively Parallelized Global Alignment Sequence Taxonomy Pipeline for Characterizing Microbiomes)
Jun 7th – 9th	Pacific Biosciences Americas User Group Meeting and Bioinformatics Workshop
Dec 1st – 4th 2014	The 9th Rat Genomics and Models Conference (Poster Presentation: JBrowse – The Rat Genome Database's New Genome Browser)
Oct 17th 2014	SAS Day Conference at the University of Wisconsin-Milwaukee (UW-Milwaukee)
Nov 2nd – 6th 2013	American Public Health Association: 141st Annual Meeting and Exposition (Accepted: Actionable Public Health Information from a Temporal Milwaukee Beach Water Quality Database; Accepted: A Zebrafish Cloud Computational Resource for Environmental Health Science Research)
Jun 30th – Jul 3rd 2013	2013 Aquatic Animal Models for Human Disease and Midwest Zebrafish Conference (Oral Presentation: A Clouded *Danio rerio* Methylation Workflow)
Jun 23rd – 27th 2013	2013 Open Science Grid (OSG) User School: Applying High-Throughput and High-Performance Computing
Jun 13th – 15th 2013	Classification Society 2013 Annual Meeting
Jun 2nd – 6th 2013	International Association for Great Lakes Research: 56th Annual Conference (Oral Presentation: Actionable Public Health Information from a Temporal Milwaukee Beach Water Quality Database)

May 3rd 2013	Biomedical and Health Informatics Research Institute: 2013 Informatics Day
Jan 15th – 17th 2013	US Geological Survey: Inter-agency Workshop to Develop Predictive Models for Beaches
Dec 17th 2012	Clinical and Translational Science Institute of Southeast Wisconsin: Genomics and Personalized Medicine
Nov 5th – 6th 2012	American Medical Informatics Association: AMIA 2012 Annual Symposium

Technical Skills

Proficient fluency in use of Bourne Shell (e.g. Bash) and Modern Perl, with additional supplemental skill in MySQL/SQLite, JavaScript, R, Python, PHP, HTML, C/C++, Java, and Lua. Comfortable with operations in Unix-compliant (e.g. GNU and BSD derivatives), Windows, and MacOS environments. Experience in working with various clusters and job submission systems (LSF, BSUB, HTCondor, and Slurm) and with the Amazon Web Services (AWS) cloud. I possess specific and significant experience with the following packages and platforms:

- Canu: a highly versatile (scales from single workstations to large, heterogeneous clusters) bioinformatics workflow for performing *de novo* genome assembly of long-read sequence datasets
- SMRT Analysis / SMRT Portal 2.3.x: an interactive, Galaxy-like pipeline execution manager for PacBio tools such as RoI / CCS1, Subreads, Methylation & Motif identification, HGAP2/3, and IsoSeq
- JBrowse: an efficient, embeddable next-generation genome browser built completely with JavaScript, HTML5, and a Perl back-end
- REDCap: a web-based framework for collecting and storing survey data in relational database format
- Trinity: a comprehensive suite for *de novo* assembly of whole transcriptomes from RNA-Seq data and downstream analysis
- Galaxy: a web-based platform for conducting transparent, reproducible biomedical informatics studies
- GROMACS: a package for performing biophysics simulations

Academic Awards, Memberships, and Honors

Spring & Fall 2015	Folk-Patrick Health Informatics Scholarship Award, University of Wisconsin-Milwaukee
Fall 2012 – 2014	Chancellor's Graduate Student Award, University of Wisconsin-Milwaukee
May 2011	Magna Cum Laude, Illinois Institute of Technology
Aug 2007 – May 2011	Camras Scholar, Illinois Institute of Technology
Aug 2007 – May 2011	College of Science and Letters Dean's List
Feb 2008	Finalist of the National Merit Scholarship Program
Jan 2009	Member of the Illinois State Academy of Science